QL Today

Volume 14
Issue 2
Dec. - Feb.
2009/10

ISSN 1432-5454

The Magazine about QL, QDOS, Sinclair Computers, SMSQ...

QL Firmware Bugs Myths - Part 2

www.QLToday.com

25% MORE pages!

"QL is 25"

Start of a new series: Tony Tebby's view of the development in computing over the past 25 years - the different paths of many different systems in the past and today!

Come to Vienna!

Details on the last page!

Contents

Advertisers

in alphabetical order

The deadline for the next issue is the 15th of February 2010. Please send material as soon as possible!

QL Today

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software
Kaiser-Wilhelm-Str. 302
47169 Duisburg
Germany

Tel. +49 203 502011
Fax +49 203 502012
email: smsq@j-m-s.com
email: QLToday@j-m-s.com

Editor:

Geoff Wicks
Flat 5b
Wordsworth Avenue
Derby DE24 9HQ
United Kingdom

Tel. +44 1332 271366
email: gtwicks@btinternet.com
email: QLToday@j-m-s.com

Co-Editor & UK Office:

Bruce Nicholls
38 Derham Gardens
Upminster
Essex RM14 3HA
United Kingdom

Tel +44 20 71930539
Fax +44 870 0568755
email: qltoday@q-v-d.demon.co.uk
email: QLToday@j-m-s.com

QL *Today* is published four times a year, our volume begins in June. Please contact the German or English office for current subscription rates or visit our homepage **www.QLTODAY.com.**

We welcome your comments, suggestions and articles. YOU make **QL *Today*** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5" disk (DD or HD) or sent via Email. We prefer ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF or JPG. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

If you need more information about the UNZIP program which is used by our BOOT program to unpack the files, we suggest that you visit Dilwyn Jones' web site where you find more information about lots of interesting QDOS software and INFOZIP at

http://www.dilwyn.me.uk/arch/index.html

Editorial

by Geoff Wicks

The party is over. Or is it?

As we come to the end of the QL's silver jubilee year, we can look back with a degree of satisfaction. The Quanta committee led the way and managed to celebrate the QL's and Quanta's own silver jubilee well within the budget it had set itself. (Support from the members was, however, a little disappointing.) Urs König secured extensive QL coverage in Personal Computer World and then organised the continental celebration. Rich Mellor was responsible for much of the similar detailed QL coverage in Retro Gamer. What other computer from the 1980s, apart from the Spectrum, could have achieved such media attention 25 years on?

This year we have looked back, but now we have to look to the future, and it is a future with changes and uncertainties.

The party is probably over for QL shows. For the second year running Quanta has been able to run only one show, and from now on the Quanta committee may have to organise its AGM and workshop itself. Last year the Italians and this year the Swiss have organised one-off shows, but, in a sense, both were nostalgia events organised by members of the previous QL groups. (However there is now a possibility of another show in Austria next year.) This year for the first time in the history of the QL there has been no meeting in the Netherlands because of health problems of the organisor. A similar situation led to the ending of the North American shows three years ago.

Workshops provided the showcase for traders, but last year two UK traders ceased active trading. Nevertheless another UK trader has exploited the new possibilities of the internet. He not only makes a profit on his QL activities, but in the last two years has provided Quanta with a quarter of its income by trading on their behalf.

The QL-users email group is also changing. Over the last few months there have been fewer detailed discussions and the group has become much more of a helpline and information point than formerly. This is a good use of the group but it is becoming more technical giving both advantages and disadvantages for its future.

Quanta has ambitious plans for its website and is working hard to realise these, but are we about to see two tiers of Quanta membership? How do you cater for those members, and there are thought to be many, who have loyally paid their subsciption for 25 years, but who are now very elderly and have no desire to become part of the internet age?

The internet may be the secret to the survival of the QL. We are an international community. The UK currently has the largest number of QL-ers, but how long will this remain so? If Quanta is to survive it may have to become a much smaller internet based organisation in which a concept of members and non-members of Quanta could become increasingly irrelevant. How long will it be before paper QL publications are no longer viable? The internet will provide the essential means for us to keep in touch with the advantage of being both cheap and instantaneous.

Change is taking place, but that does not mean there is no future for the QL. It is how we adapt to that change that could determine the nature of the QL's survival.

NEW SOFTWARE DATABASE

QL trader Rich Mellor has started an ambitious project to catalogue and preserve as many commercial QL programs as possible, a task he sees as important to ensure the continued future of the QL. Where possible the intention is to provide the programs in a suitable form to run on one of the PC QL emulators. It will also be a resource for QL users who have a legal copy of a program that will no longer load.

In Rich's own words:

"We have managed to get most of the software from microdrive onto a PC in a format for use with Q-emuLator in the main, although it does help us to make fresh working copies on microdrives and disks. The idea behind this, is to ensure those users who have an original copy, but cannot get it to load anymore, will be able to purchase a working copy on disk or microdrive, or for use with Q-emuLator from our website (provided that they can prove ownership of the original program).

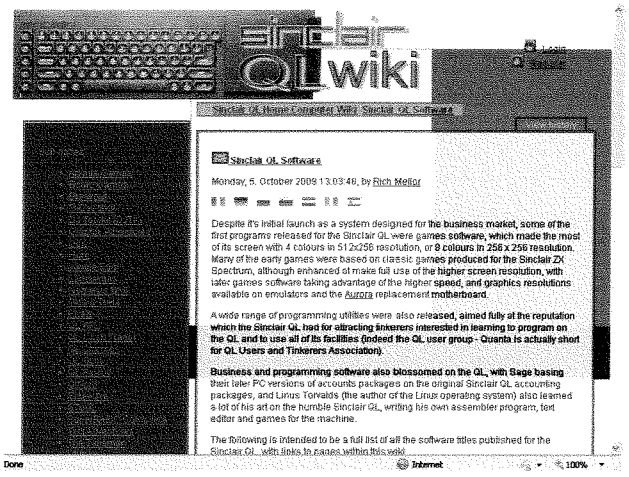
As part of this work, we are also updating the QL Wiki to include more information on the software, hardware, books and traders that have been around since 1984. We would welcome more input into the traders and personalities section in particular if anyone knows the history of any of the software and development houses (however short lived!).

Lots of work to do in updating the software details, but we hope that this will address one of the main issues when people talk about the Sinclair QL - namely they are uncertain what software was produced for it!!"

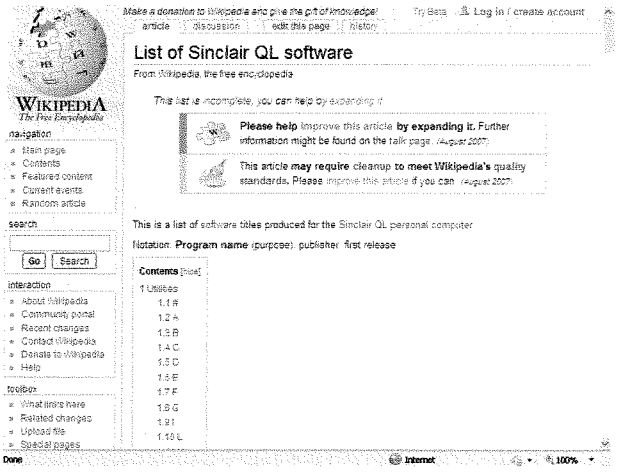
First reactions to Rich's announcement showed that there was some confusion about the wiki. Several QL-ers assumed that he was referring to the QL section on wikipedia and not to the dedicated QL wiki on his own site. Rich set this up early in 2007 mainly as a software resource to give more QL information than was possible on wikipedia and later expanded it to a more general QL wiki. The launch of the wiki was reported in QL Today together with a review (Vol. 11 issue 5).

As several QL-ers pointed out the wiki still has some serious omissions. The wiki concept is that almost anyone can contribute, but so far Rich has had to do the bulk of the work himself. The wiki is to be found at:

www.rwapadventures.com/ql-wiki/



Rich has also placed the software database on wikipedia.



The QL has a second dedicated wiki set up last year by QL Today writer, Norman Dunbar. This is biased more to the technical and programming side of the QL and can be found at:

www.qdosmsq.dunbar-it.co.uk

Rich has been able to re-release some QL games, one of which was by popular request:.

"Due to popular demand, and with the consent of Jochen Merz, we are pleased to announce that we have re-released an old QL arcade game, Pengi, which was written by Jochen Merz and previously released on the Gigasoft label.

In this colourful game, you control a small Penguin (Pengi) as he tries to survive the Antarctic climate. Faced with deadly snobees all around, the only way to survive is by pushing ice blocks onto the snobees and squashing them, before they touch Pengi. You can also collect diamonds to gain additional points."

This increases the range of arcade games still available commercially from RWAP Software - **http://www.rwapsoftware.co.uk/games.html**

Rich has also re-released Cuthbert in Space and QL Hopper.

Other programs that he is attempting to rescue include Viewpoint by Rubicon, Concept 3D by Tesseract Software, QL Gardener by Gordian Computing Services and CAD PAK by Datalink Systems. As a teaser he adds that Britain's main consumer protection organisation, the Consumers Association, once released software for the QL.

Finally a piece of late news from RWAP They have launched a new website: **http://sellmyretro.com**

MORE SOFTWARE

DILWYN JONES

Once again Dilwyn has a long list of new and updated items.

ZIP MANAGER

I've added an update to my Zip Manager program to the Archivers page on my website.
Version 1.06 of this program fixes a bug in the Delete command, which sometimes got confused between '.' and '_' filename extension separators.
It's available to download free from
http://www.dilwyn.me.uk/arch/index.html

MIRACLE MIDI INTERFACE

Miracle MIDI Interface - for anyone who acquires a Miracle MIDI interface for the QL without the software disk (a program called Tracker by Dan Gaffey), it can now be downloaded from the Misc software page on my website. Note that the software can only be used with the Miracle MIDI interface, it cannot be used with any other interface. I am grateful to Derek Stewart for locating a copy of the software for me.
http://www.dilwyn.me.uk/misc/index.html
A more general purpose free MIDI software package for a QL is Al Boehm's Midi Player 2 package, available from
http://www.dilwyn.me.uk/sound/index.html

PCB CAD

I've just added the latest update to Malcolm Lear's PCB Cad program to my website. Here are details of revisions made since the last version available (6.63):

6.64 Corrected minor errors in SMD1.lib library.
Corrected operation of window resizing.
OUTLN is just optional.

6.65 16-09-09 Layer names can now be changed.
Dark white which was displayed as mid grey now changed to light grey and dark black which displayed as black now changed to dark grey.
File access tests using DEVICE_STATUS changed to be compatible with older QL systems.

Later a further upgrade to version 6.67 was announced.
The program is a 1.22MB download from the Graphics page on my website:
http://www.dilwyn.me.uk/graphics/index.html
As late news as QL Today was finalising the news pages, Dilwyn announced the first pointer version of the program.

DIGITAL C SOURCE

With grateful thanks to the author, Gerry Jackson, the source files for the Digital C SE compiler system are now available from the Languages page on my website.
Once you have downloaded it, please read the README.HTML file it contains.
http://www.dilwyn.me.uk/language/index.html

EDDICON AND SPRLIP

Two new free programs from Duncan Neithercut were recently added to my website.
Eddicon is an icon/sprite editor to create sprites in mode 64 or mode 4 or to save them as .bmp for use in Wolfgang Lenerz program. Existing sprites in a variety of modes such as 32 and 33 (QXL/QPC and Q40/Q60) can be loaded and edited. Or sprites can be created from scratch. The editor has a number of novel features including an independently editable alpha channel, undo function and a simple merge to combine 2 sprites of the same size, and other features such as home directory and colour theme awareness. It is in an alpha/beta status but is fully usable, but there may still be bugs due to the complexity of the program, that require additional users to identify.
Complementary to Eddicon is a program to concatenate a list of sprites into a single library file that can be loaded by LRESPR or linked into a Qliberated program using the REMark $$asmb directive. The addresses of the sprites in the library may be accessed through a single keyword that is part of the library file. This makes it straightforward to add multiple high colour sprites to Qliberated programs. Eddicon uses this system for its mode64 icons although the sprite library maker program will work with sprites of any mode and a mixture of modes in a single file.

Both packages may be downloaded from my website at
http://www.dilwyn.me.uk/sprites/index.html

GEORGE GWILT UPDATES

George has announced the following updates to his programs:

I have put on my site amended versions of NET_PEEK and GWDISS. The amendments correct the disassembly by GWDISS of some ColdFire instructions and add the disassembly of all ColdFire instructions to NET_PEEK. Without this change the previous version, 3.39, of NET_PEEK could crash while disassembling.

A new version of UCONFIG which produces config blocks for S*BASIC, Assembler and C programs. The new version allows a full alteration of an existing config block from the input of a _INS file (which is the config block for an S*BASIC program). The output is a full set of files for all three types of program S*BASIC, Assembler and C.

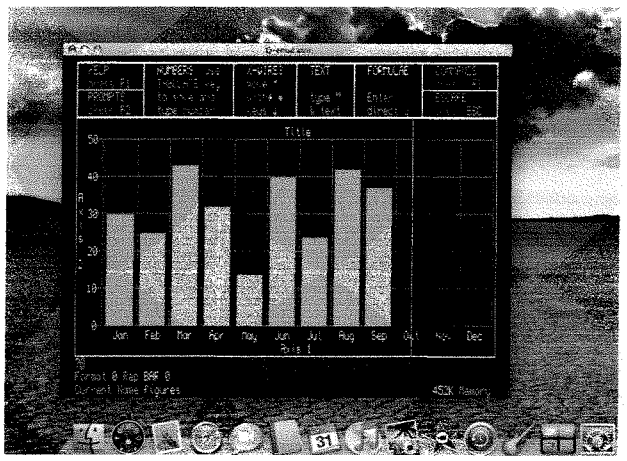
I wrote the new version because I was fed up with having to reproduce from scratch all the old information every time I wanted to add an item to an existing block. This might therefore be of use to other people as well.

http://web.ukonline.co.uk/george.gwilt/

OS X Q-emuLator VERSION

Daniele Terdina has started work on an OS-X version of Q-emuLator although he says that it is still far from completion. However he has released a screen shot. More details can be found at:

http://www.terdina.net/ql/MacQL.html



SGC BATTERIES

Several QL users are finding their Super Gold Card battery is running low and supplies of the original battery are now unavailable. A symptom of a low battery is an inaccurate time and date when switching the machine on.

Davide Santachiara has offered some help with this problem:

"I replaced two batteries both on a GC and a SGC of a friend of mine thanks to a standard CR2032 3V lithium battery which the Italian QL hardware expert Romaldo Parodi, was able to get with presoldered pins on it. Then it was quite easy to solder a small additional wire to reach the two pins where the original battery was placed (actually four pins are present but only two are used).

I have shot some pictures of the procedure, if somebody is interested in getting them just email your private email address."

d.santachiara@libero.it

WEBSITE MOVE

There are persistent rumours that Geocities, who host Davide's webpages, is shortly due to close. Davide has moved the webpages to:

www.sinclairql.it

The site hosts the Ergon Development software including the Spectrum emulators. At the time of Davide's message he had not checked that the new site was fully working correctly, but hoped shortly to do so.

ONLINE MANUALS

Following the Swiss QL meeting Marcel Kilgus has placed the "QDOS SMSQ Reference Manual" and the "QPTR" manuals online in PDF format.

http://www.kilgus.net/smsqe/development.html

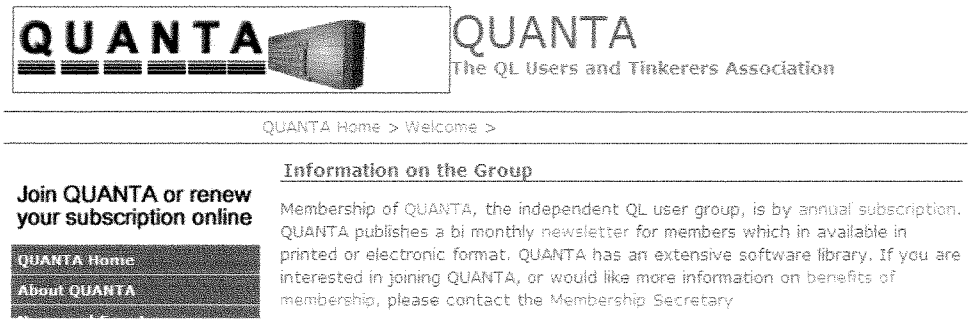
WITHIN BUDGET

Quanta has celebrated its silver jubilee year well within the budget it set itself. Using the 2005 QL is 21 show costs as a guideline it had budgeted £3,500 for the silver jubilee events. Provisional figures show the actual cost to have been £3,000 and as icing on the cake this figure includes the £572 cost of the special silver jubilee issue of the Quanta Magazine that had not been included in the budget.

Quanta has also been looking at the future of its website and in particular at systems for expanding it and ensuring up to date content. At a recent committee meeting two systems, Joomla and Typo3, were demonstrated and the committee opted for Typo3. The final content of the website and the division between public and restricted content has yet to be decided. It is also not known how quickly changes will be made to the site.

Members can still pay their subscription via the website and PayPal. Some members have experienced difficulties doing this in the past. The link -

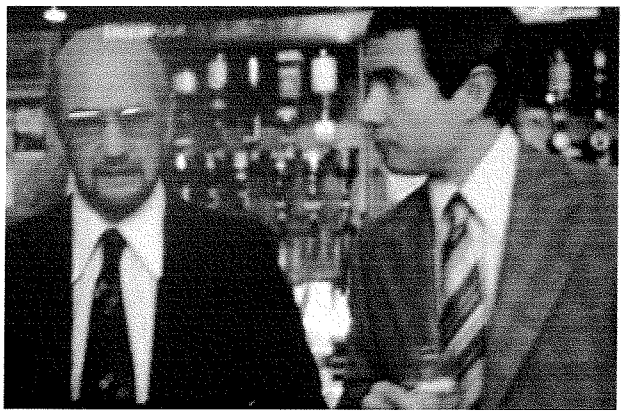
on the top left corner of the home page - was seen by some as being an advertisement for the facility. This is an animated graphic showing three different pictures. In our illustration it has the message "Join QUANTA or renew you subcription online". Alternatively click "Quanta Magazine" on the home page menu and then "online subscriptions". During the transaction PayPal places a cookie on the computer and users with high security settings may have to reduce these temporarily.



For the second year running Quanta has only been able to run one show. Next year it plans to hold the AGM in April at a venue in the Midlands.

MICRO MEN

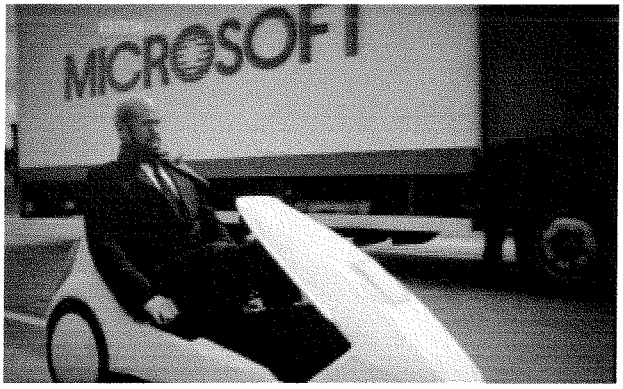
Although not a part of the QL's quarter centenary there was much interest in a BBC television play, Micro Men. This dramatised the 1980's battle between Clive Sinclair and Acorn Computer's Chris Curry - formerly Sinclair's right hand man - to obtain a lucrative contract for the BBC school and home computer. Partly based on fact and partly on fiction the play left the QL viewer frustrated by giving little indication of which bits were true and which made up. Clive Sinclair was portrayed as an ill-tempered man obsessed with developing an electric car and whose fatal weakness was an un-



willingness to listen to others. Chris Curry came over as the better businessman and boss who won the BBC contract. However it was not all plain sailing and one delightful scene showed him stalling the BBC officials while his staff were desperately tweaking the prototype machine to make it work.

Later scenes showed a frustrated Clive Sinclair in despair because his computers were seen as games machines and Chris Curry jealous of Sinclair's success in capturing the games market. Sinclair went upmarket and this is where the QL made an appearance, and Curry downmarket with a cut down version of the BBC computer, the Electron, as a games machine. By now the boom was over, the QL was a commercial failure and disaster loomed for Acorn. Curry took a massive order from a high street retailer, but failed to get it confirmed in writing so that he was left with a warehouse full of 120,000 unsaleable Electrons.

The drama ended rather cruelly with Sinclair peddling along the road in his C5 and being overtaken by a Microsoft juggernaut.



The play was a mixture of fact and fiction, but for a detailed and accurate account of the QL's development the best place to look is the current volume of QL Today.

QL Firmware Bugs Myths - Part 2
Sinclair creates the myth

by Tony Tebby

The press reaction to the launch was enthusiastic or even ecstatic, as shown in the articles that appeared in February. Within Sinclair, foreboding was a better word for the mood. It was five days before I found out what was wrong: 28 day delivery had been promised, the hardware was far from ready, SuperBASIC was untested and had only a minimum of procedures and the state of the Psion suite was a complete unknown.

I immediately went to see Sir Clive to tell him that I would stay on only until QLs were delivered to customers. Then I cornered one of the directors to complain about the launch. He explained to me that the commitment to 28 days delivery had been necessary as a number of senior Sinclair personnel had bought Sinclair Research Limited shares when Sir Clive sold his 10% and, with the various delays to new products, their shares had lost lot of their value and they had no choice but to launch the QL and take orders to boost the share price. This sounded to me to be rather dishonest, but the director concerned considered it to be rather astute.

When QLs were not delivered within 28 days, the brown stinky stuff hit the fan. A journalist, Guy Kewney, who had not been taken in by the launch (he knew an empty box when he saw one) was very suspicious. He wrote an article explaining that if a director of a company made a false statement to manipulate the share price of that company, then this was fraud. When news of this article reached Sinclair, the chaos transmuted into panic: clearly someone had leaked the truth to the press – the directors could not believe that a journalist could have worked it out for himself. I received threats from the directors of the company. I found this more offensive than worrying. Despite the dishonesty of certain directors, and the weakness of others who refused to stand up to them, Sinclair Research Limited was my employer and I owed the company my loyalty.

What did happen

I can only give the true story for that which concerned me directly. At that time I had to assume, and I still have to assume, that anything I heard about the progress on the QL from anyone else at Sinclair (other than Jan Jones who was with me) was at best mere rumour. Moreover, word was out that talking to me was not a good idea if you wished to continue your career at Sinclair (I had already resigned so I was now an outsider). This made project coordination tricky. I moved out to the Milton Hall building site (the future Sinclair Laboratories) with Jan Jones.

One of the well founded stories was that when the QL was launched with a promise of delivery in 28 days, there were no working prototypes. I think I can confirm this. I had never seen one, and some time later I was asked by one of the directors "when will there be a stable version of the software?" I think my reply was along the lines "about a week after we have a stable specification and a stable hardware platform" - "Ah, in that case when can we have a test version" - "About a week after I get a prototype" - "You must have a prototype, the machine has been launched". It appears that I was not the only naïve person around.

At which point, this director proved to me that there was at least one senior person in Sinclair who was not panicking. He managed to obtain all the bits necessary to build a QL. I obtained a wooden panel and some screws from elsewhere in the building site and assembled the bits rather like an exploded diagram. Amazingly, it showed signed of life when I fitted an EPROM set (to the external EPROM card that was later to turn into the infamous "dongle" or "kludge") and turned the power on. This was THE prototype machine.

But I soon understood why it there was no other full working prototype. When the power supply was delivering enough current to keep the computer working, the ripple was so great that the on-board regulator dropped out. I changed the power supply and added large smoothing capacitors. (By using my own power supply and having an open QL, I missed out on the exciting story of increasingly powerful power supplies, on-board regulators that overheated if you put the QL PCB in its case, the machine

crashing when a Microdrive motor was started, etc.). I then poked around a bit before adding lots of decoupling capacitors. This seemed to improve the stability.

So, I had my prototype. Did I manage to finalise the operating system and drivers in a week, so that the system could be tested for another week before committing the software to ROM? Not quite.

The Microdrives and the "Notwork"

The first Microdrive problem had cropped up a month before the launch, before there was a full prototype.

Microdrive / network problem 1

I had been told that there was a slight problem on the first PCB layout. It was more than a slight problem. Several months before, when the block diagram for the ZX83 was being converted to a complete logic diagram, the data lines on the future ZX8302 chip were connected to the RAM bus instead of the processor data bus. The ZX8301 glue logic was also designed with the ZX8302 enable and handshake signals derived from the RAM bus timing signals and not the processor bus timing signals. Why was this important? The RAM bus is shared with the display which steals a block of 8 memory cycles out of every 12 cycles during display lines. The ZX83 relied on the processor for critical timing operations that would be carried out in hardware on more normal systems. You cannot perform critical timing operations if you cannot determine how long each instruction will take. The timing "jitter" for this design could be reduced by ensuring that the instruction loops accessing the ZX8302 were a multiple of twelve cycles long. For the Microdrives this was possible although it reduced the margins a bit. For the network, however, the nearest to the Spectrum timing was 7x12=84 cycles – not close enough. In addition, the residual jitter (2.5μ), on both transmit and receive, gobbled up half the timing margins. The network could be neither Spectrum compatible nor reliable.

Microdrive problem 2

When I finally had my own hand-built prototype QL, I was anxious to try out the fixed Microdrive drivers. When I tried the Microdrives for the first time (now at launch + 3 weeks, I think) the drivers did not work at all. Format a Microdrive and all you got was "format failed". The code was timing critical so it could not be traced, and after spending a lot of time hunting through the code for potential errors, I tried a scope. Looking at the signals I thought that it might not be a QDOS (the new name for Domesdos) driver bug, it might have had something to do with the signal to noise ratio from the Microdrive head amplifier being 0 – no signal, all noise. Ben Cheese provided me with a fix – a capacitor to be soldered onto the Microdrive PCB. Another QDOS bug fixed – without changing a single byte. But why did my, straight from production, Microdrives not have that capacitor?

Microdrive / network problem 3

And then the next problem with the Microdrives. The driver sometimes lost data – it seemed that it was just not fast enough for the job. I had calculated the execution time of the Microdrive read routines down to the nearest cycle. I could not see where my calculations were wrong. I struggled with this for a couple of days before Jan, who had no electronic design experience, came up with the solution that had eluded my obviously very softened brain. "Why does the Microdrive only work reliably in the morning?" she said, all innocent like. Ah! The real question was "Why does the Microdrive stop working when the QL has been on for a little while?". I set off to Sinclair's labs and came back with a can of freezer spray. It did not take long to discover that, as long as there was a nice coating of white frost on the ZX8301 chip, I could read data reliably from the Microdrives. But the ZX8301 chip has nothing to do with the Microdrives, does it?

It turned out that there was a timing race in the ZX8301 chip design: when it warmed up, the RAM bus contention circuit failed to register a request for access to the bus until it was too late. This problem would have probably gone undetected, for the whole life of the QL, had it not compounded the ZX8302 access timing fault described above: the 12 cycle workaround could never work on a warm QL. Fixing the hardware properly would take several months for a new ZX8301 design to get into production and the QL was due for delivery the next week. Unfortunately the hardware workaround, which involved wiring the data bus directly to the ZX8302 pins and adding another spider (effectively the D14 build solution), would not solve the problem completely: although the processor would still run more slowly when the ZX8301 was hot, this would only occur when it was accessing RAM.

The software workaround for this double ZX8302 timing fault was to slice one or two cycles off the ZX8302 read/write loops. This worked more or less for the Microdrives, but it eat up another half of the timing margins for the network: the Spectrum team renamed it "The Notwork". However, all would be fixed on the next version of the PCB (joke).

Microdrive problem 4

And then the next problem with the Microdrives. All of a sudden, I got a report that Microdrive access was very slow (several minutes to read a modest sized file) on the first pre-production QLs. I tracked this down to an unnotified modification that had been made to the Microdrives themselves. The original specification for the Microdrive had the total tape stop and start distance less than the length of a sector. The original aim of the device driver was to minimise the time for which the drive was running to minimise the tape wear and power consumption (the ZX83 was supposed to be battery powered, remember). The sectors were laid out with a one in two interleave so that if the drive was stopped after reading a sector, the tape would stop on an "uninteresting" sector and then the next sector to be read would hopefully be the next sector of the file. Once a file had been found, access was relatively fast. However, in a vain attempt to prevent Microdrive cartridges self-destructing, someone decided to try slugging the motor turn on and turn off – increasing the total stop and start distance to nearly 10 sectors.

High speed photography showed, however, that the acceleration and deceleration was unaffected, the modification merely delayed stopping and starting, doing no good at all – but plenty of harm. Firstly, it greatly increased the running time of the Microdrives and, therefore, tape wear. Secondly it ensured that, for a stop/start scenario, the next sector would be missed unless it was at least ten sectors away, reducing serial read speeds by a factor of five. Thirdly it increased the surge current when the Microdrive motor was turned on (current surge => voltage dip => QL crash).

ICL rejected this modification for the OPD and it was not tried on the Spectrum (see Wikipedia "ZX Microdrive"). So why was this purely harmful modification retained? Was it sheer idiocy or was it a deliberate attempt to delay the release of the firmware?

I had to spend most of a week rewriting the Microdrive drivers to implement a radically new strategy to claw back some of the performance lost by this modification. "the software is still not ready for shipping and we are well past the 28 days". Even with the rewritten drivers, the performance of the QL Microdrives was still well below that of unmodified Spectrum Microdrives. Moreover, in the rush, there was an serious oversight: the QDOS memory manager assumed that the filing system could work with only one buffer. The new pre-fetch strategy required a minimum of two buffers. Oops.

Other problems

Of course, there were many other hardware problems that I had to deal with. This should be considered normal. If there is a software workaround for a hardware fault, this is almost certainly cheaper and quicker than a hardware fix, although it will often be a less than complete cure.

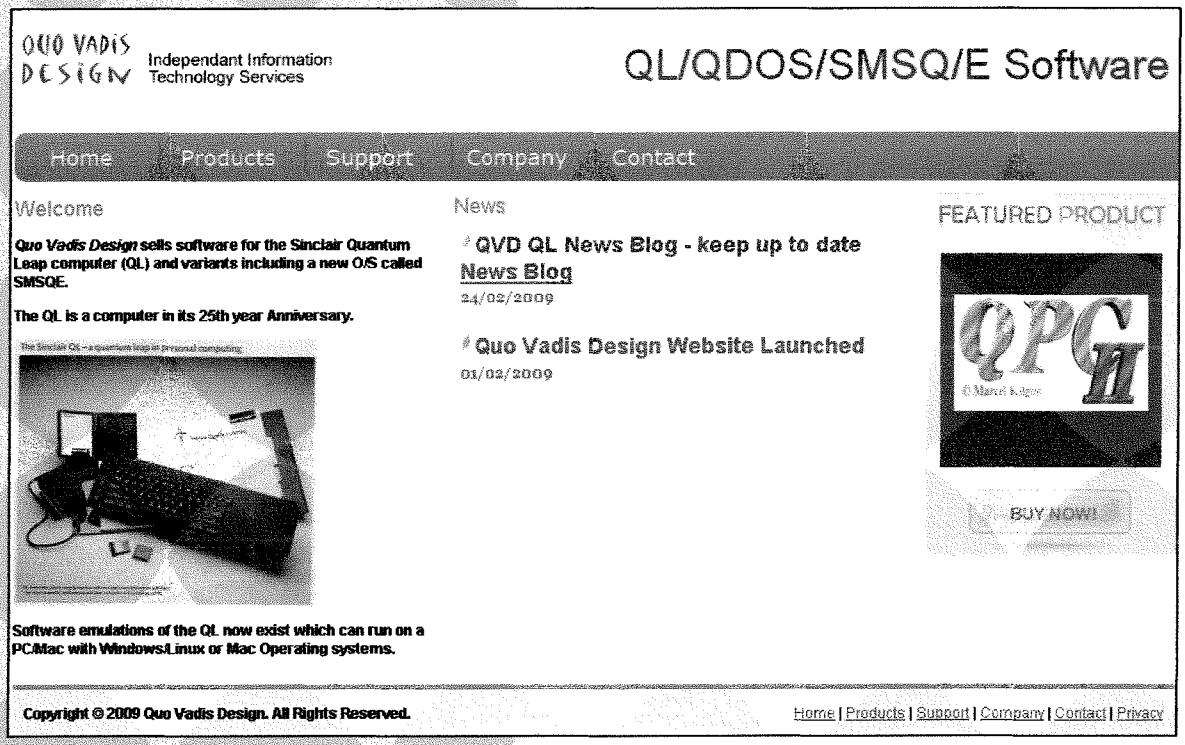
Meanwhile, back on the farm

After the launch, Jan Jones was busy trying to work out what she should put into the SuperBASIC for the QL and, possibly more important, what to leave out. With many regrets "WHEN" fell by the wayside – that would shave a good month off the timescale, but there was no shortage of suggestions for the priorities for adding Spectrum compatible features. Jan, therefore, took the very reasonable course of "code it quick and test it". After the first totally untested version (FB) was made available for independent testing, the bug reports and "wish lists" came flooding in. As I remember it, in addition to coding new features on the fly, she was also clearing up to 20 bug reports a week and new versions were made available for testing almost weekly. The shakedown was rapid with less than one new bug for every ten new functions. The flow of bugs reports dried up when we made AH (1.02) available for testing and we called a halt to further changes to SuperBASIC. Starting with AH, we changed over to "code inspection" (reading through all the code written over the previous 9 months looking for potential, or real, problems.

QUO VADIS DESIGN

Independant Information Technology Services

www.ql-qvd.com



Bruce@ql-qvd.com

Quo Vadis Design
38 Derham Gardens
Upminster
RM14 3HA
UK

Tel: +44 (0)20 71930539
Fax: +44 (0)870 0568755

Special Offers available from Jochen Merz Software for its 25 years in QL Trading

Check the QL News Blog on our website for updates. www.ql-qvd.com/blog

The final shakedown

Numerous "bug lists" have been produced for the QL firmware. These show a number of changes between AH and JM (1.03).

1. A "not-a-bug" was found in the SuperBASIC interpreter by code inspection. Floating point arrays were limited to 65536 elements. As there were no QLs with enough memory to store an array of 65536 floating point numbers, this limitation had not shown up in testing.
2. A serious bug with workaround was found in the OS core by testing. Opening a channel to a file that was already open left the file pointer pointing to the start of the "distributed directory entry" in the file and not the start of the data in the file. This was an oversight made when patching the filing system to improve the recoverability of files on a damaged Microdrive cartridge.
3. A serious bug (no workaround) was found in the SuperBASIC interpreter by code inspection. String comparison in SuperBASIC compared numerical values embedded in strings by decimal value (it was to take Microsoft a decade to catch up with this, while UNIX still has not even managed to get to grips with upper and lower case). Unfortunately the SuperBASIC string comparison thought that "." was the same as 0 (as well as 0.0, .0, or 0.).
4. A fatal bug (no workaround) was found in the OS core by code inspection. Opening a channel for a job that does not exist crashes the system. Readers used to the extremely restrictive UNIX and Windows environments might wonder how a job could open a channel for another job and why you would want to do it. There is a good reason.
5. A fatal bug (no workaround) was found in the SuperBASIC interpreter by testing. The buffer handling for INPUT data was less than ideal and if a line longer than 128 characters was typed (the default buffer length) bizarre things could happen.

A week's independent testing of version AH had thrown up 2 bugs and a week's code examination had identified 3 bugs (according to the best lists I can find. I am sure that there was a third bug in the AH OS core that I found and fixed, but I cannot remember what it was and it must be so obscure that no-one has ever reported it publicly).

By this time, no-one was interested in delivering SuperBASIC as a base version with extensions, so the JM version, with the complete set of procedures and functions and all the graphics, was handed over for committing to ROM. This must have been sometime in March. Code examination and testing of continued, but no more changes were made to the release version and, over the next couple of weeks, nothing was found that would have merited recalling JM and replacing it by TB, the next stable version.

Subsequent user experience has shown that, apart from my monumental error of patching both Domesdos and SuperBASIC to get them to fit together, JM was fairly sound and the bug list built up over the first year, while not as good as it should be (no bugs at all), was quite respectable for software finished under what might be described as difficult conditions.

With JM released and TB "standing by" for any corrections, we started a new development series with the JS version. Jan had started working on adding the WHEN constructs when we heard that QL shipments had started and that customers would soon be receiving their long awaited QLs. My job finished, I notified the personnel department that I had at last gone and moved out of Milton Hall into a Portakabin (hired by Sinclair, on Sinclair's property), in front of the main entrance, where I could be reached in case of emergency.

The start of the myth

The next is partly hearsay, I was out of the loop, but it seems to be confirmed by the rather guarded reports I have seen by the journalists present.

When the first QLs were ready to be shipped, Sinclair Research organised a press demonstration. It is not clear what the purpose of this demonstration was. It certainly made a very bad impression on the press. That may or may not have been the intention of those who organised it.

The QLs on demonstration had a black dongle hanging out of the back, as did the first QLs shipped to customers. These QLs were also equipped with pre-test OS and SuperBASIC, without all the work-arounds for the known hardware faults, with their own known bugs, with only a subset of the Spectrum procedures and with some procedures having different parameters from the final version.

Having a dongle hanging out and having pre-test software are strange features for machines intended to impress journalists.

The dongle

Even if you find it difficult to believe that the JM version was available from well before the launch, the dongle itself would have been strange. The reason most often quoted, that the firmware had turned out larger than the planned 32 kbytes and so the PCBs had to be modified to take larger ROMs, was totally untrue.

1. There never was a planned size for OS + SuperBASIC because this was never planned.
2. From the start, all PCBs had the wiring for up to 64 kbytes of ROM.
3. From January, all PCBs had two sockets specifically for 32+16 kbytes of ROM.

The QL PCBs were, however, designed to take ROMs with on-chip address decode. EPROMs cannot be fitted directly into the QL PCB: a spider is required for the address decode. But if, for the sake of argument, we take it that that shipping QLs with EPROMs rather than ROMs was a justified technical solution, there is still no justification for the dongle. Why ship with three 16k byte EPROMs rather than a 32k byte EPROM and a 16k byte EPROM? A 32k byte EPROM did not cost more than two 16k EPROMs + a PCB to plug into the ROM socket + case + the assembly costs. Even mounting three 16k EPROMs internally (as in build D06 where three 16k EPROMS recycled from returns were used) would not be more expensive than using a dongle on the outside.

When I found out about the dongle on the QLs that were being shipped to customers, I contacted production and was lucky enough to get hold of a friend. Apparently, the dongle was the outward sign that the QLs that were being shipped were "pre-production" (i.e. did not pass even basic tests): they would all be recalled and replaced as soon as fully functional QLs were available. Being responsible for quality, he thought it was stupid, but it was a decision from above. He was unaware that the FB version being shipped was not the current, tested, released version of the firmware. He had been told that the later versions had not been tested and might be less reliable than FB. There was, apparently, internal disinformation as well.

The story about the dongles I heard from production seems to be confirmed by other sources. The Sinclair service manual for QLs only covers builds with ROMs or "piggy-back" EPROMs (build D06 onwards): dongled QLs were scrapped automatically as it was not considered possible to rework them. Elsewhere, builds up to and including D05 (pre issue 5 PCB) are referred to as pre-production and there are no "mandatory modifications" for these to bring them up to standard – they are just scrap. So the dongle really was just an excuse for recalling non-working QLs to be scrapped – despite the well publicised policy of "recalling for firmware upgrade".

Pre-test software

Having established that production did not have the current version of the software, I did then manage to get hold of a director who was apologetic but he explained that the press would accept software bugs more readily than faulty hardware (it turns out he was right, the press swallowed the story hook, line and sinker). This had nothing to do with whether there was a dongle hanging out of the back or not, although the dongle certainly helped. Faulty software was being delivered deliberately as a smoke-screen.

The demonstration

However, if you were organising a press demonstration and wanted to make a favourable impression, would you provide machines with a dongle hanging out of the back? What sort of deranged person would think that pre-test software would help dispel the conviction that the QL was "not fit for sale"?

The report in the June issue of Practical Computing summarises the event well.

"The bad news is that QDOS and the bundled software's current implementation is what one of Sinclair's engineers described as 'flaky'. Even basic operations like retrieving specific bytes from Microdrives brought the system down. Several of the bugs thrown up in the session seemed new to Sinclair, and were noted with bemused interest." There are several things about this report.

1. The firmware was described as "flaky" by only "one of Sinclair's engineers". The others did not. Even the term flaky is revealing. It is not a description of software, it is the denigration of a person: "an offensive term describing somebody regarded as eccentric or irrational" or "a procrastinator, a careless or lazy person, dishonest and doesn't keep to their word".
2. "Retrieving specific bytes from Microdrives brought the system down": no such firmware bug has ever been reported on any bug list that I have seen. There were, however, serious hardware problems.
3. Only "QDOS and the bundled software" (the Psion suite) were mentioned as being bug-ridden. Later reports only mentioned SuperBASIC as suffering from bugs.
4. "Several of the bugs thrown up in the session seemed new to Sinclair, and were noted with bemused interest". For bugs in the Psion suite this would not be surprising, as I do not think anyone at Sinclair had seen the Psion suite in action on a QL. As far as the firmware was concerned, those who had taken part in the independent testing up to JM, would certainly have been bemused at seeing bugs that had been fixed weeks before.

At this demonstration, did anyone from Sinclair actually say anything that was untrue? The firmware in the QLs on demonstration was unreliable – it was the first test version – but did anyone from Sinclair say that it was the current version or was this a reasonable, if incorrect, assumption by the press? Did anyone from Sinclair say that the PCB could only take 32 k bytes of ROM, or was this assumed by the press because there was 16 k bytes in a dongle hanging out of the back? It seems that it was not necessary for anyone to lie about the state of the software development: the press was quite able to make up its own stories.

This demonstration with dongles and pre-test software was, however, only the start of the disinformation campaign. It was certainly not casual misinformation: over the next few months, the firmware versions that had been produced at weekly intervals in February and March 1984, appeared in production QLs at monthly intervals to give the false impression the software was being updated all through spring and early summer of 1984. Who was updating the software? Sinclair had no-one working on QL software until after the summer recruitment of new graduates.

This was clearly not enough mask the real problems. By mid summer, the first stable version JM was being delivered, but the hardware was still not fully functional and some journalists had noticed that software updates "cannot make any difference to the hardware faults".

Was this smokescreen just the result of panic in a company that had fraudulently promised 28 days delivery, had taken the money and, three months later, found itself still unable to build working machines and desperately needed to cover up the truth, even if the cover-up destroyed the company? How many people were involved in the decision to ship pre-test software in place of tested software? How many even knew that there was fully tested software ready to ship? Was someone taking advantage of the panic to try to turn the situation to their own advantage?

Reasons for the smokescreen and the nine month delay to issue 6

After I wrote the first draft of this (true) story, I managed to contact some of those on the periphery of the project. This provoked some spontaneous comments of the nature "They can't do anything to stop you now" (who are ***they*** and why would ***they*** want to stop me from doing ***what***?) and "It's about time someone said how you got shafted". They also enabled me to fill in quite a lot of the pre-launch story but they suffered from near total amnesia for the critical period of the six months after the launch. One of the amnesiacs told me that he did not know what happened because, at the time, he decided that he did not want to know what was happening.

What they agreed on was that the panic was total. Sinclair had angry customers, the Advertising Standards Authority and the Office of Fair Trading all leaning on it. Admitting that the working hardware could not be delivered three months after the launch would be equivalent to admitting that the hardware was not working at the time of the launch and that Sinclair had taken money without being able to deliver – and that would add criminal investigations into the pot (about two years later, Chris Skogland of Medic was sentenced to 6 years prison for "reckless trading": being the director of a company that had taken money for a QL accessory that was not ready to ship). Delivering QLs made it possible to justify taking customers' money, even if the QLs did not work.

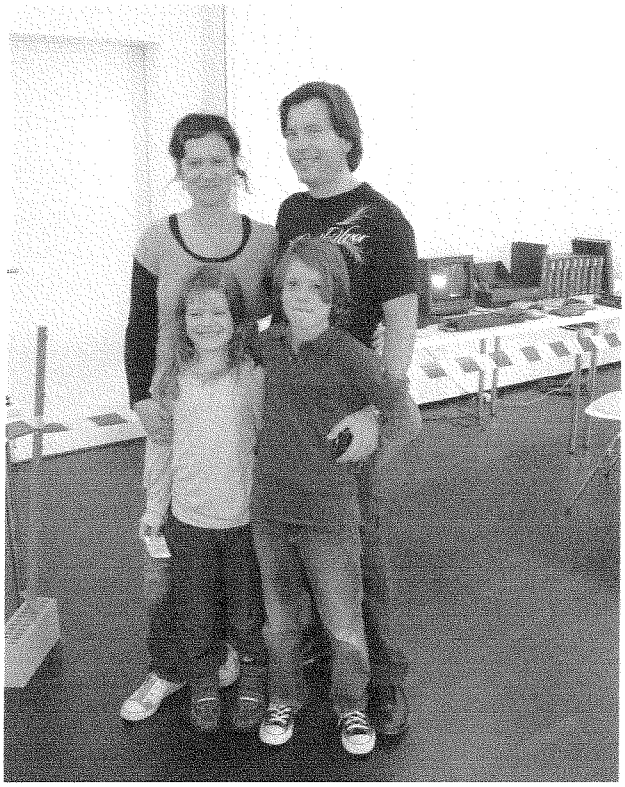
A convincing explanation for the nine months it took to ship fully working QLs also emerged. All the modifications required to make the machine reasonably reliable were known by March, but, apparently, there was a total refusal to "delay" production to fix the problems, so, for months, production of non working, or barely working, machines staggered on in the hope that the customers would accept the QL as it was. There were directors completely unable to accept that, with a £400 price tag, the QL had to be, not just as reliable as, but significantly more reliable than the Spectrum.

QL and Mac are 25

by Tony Firshman

31st of October and 1st of November 2009 - Luzern, Switzerland

When I first heard of the show in Lucerne (Luzern) that Urs König was planning in Lucerne (Luzern) I knew this was another good excuse for a short tax-deductable 'holiday'. However there appeared to be no-one coming from the usual flock of QL traders. However Simon Goodwin fancied the idea, and a group of the usual villains (Dilwyn & Ann, Jochen & Andrea and Marcel) joined in.



Urs König & Family - thanks for organising this event!

I offered Simon a lift to Luton airport on my motorcycle - one gets **free** parking in the short-term car park! I was amazed he accepted as he burnt and destroyed a shoe last time he rode pillion with me. Simon though wanted to see Ian Pizer in Geneva as well - 200 miles or so from Lucerne. I worked out the cost of hiring a car, and it proved cheaper, allowing for a free stay at Ian's for both of us, than us going separately by public transport. Having avoided the Villa Marias in Czechoslovakia and Germany, I managed to book, by email, rooms in the excellent looking and cheap guest house "Villa Maria" in Luzern. Despite her really awful English, and seemingly stuck capslock key, it all looked good. I also booked Easyjet **and** car hire very simply. The car hire, at £30 a day was especially good value.. Even better, Dilwyn and Ann were on the same Easyjet flight, so we negotiated a 'taxi' fare to Lucerne. Switzerland have a brilliant tourist trap. Motorways require a cheap £20 annual vignette. Tourists have to pay the same, but of course only for the duration of their trip. .... or so we thought (see later).

We arrived at Luton in good time, especially as we bypassed the good 30 minute car queue into the airport. We had hand baggage only, but my case strapped on my top box along with me and Simon must have been quite a sight. On arrival at Zurich, I could go straight to the car hire while Dilwyn and Ann had to wait for their hold baggage. All was fine until we tried to escape from the airport. I had my motorcycle satnav on the windscreen (using a suction pad). That was great but airports do not have addresses so the way out was a mite unclear. We only had to navigate out of car hire and out (using the free car park ticket) twice! It was complicated by the satnav taking a **long** time to find all those exciting new satellite fixes. She (the Garmin voice) seemed overjoyed announcing that we were in Switzerland. I resisted Simon's offer to find a German voice! We did though set it to kilometres which would help with speed limits (see later). I must say I find 'exit in 300 metres' an awful lot more comprehensible than 'exit in point four miles'. Why on earth don't Garmin use yards like TomTom!

It was very appropriate that the emergency pack in the boot of the car was branded *König*.

Anyway the "taxi" eventually set out with the Welsh tourist fare to Lucerne. I had stored the relevant routes in my Satnav and all was fine. I had forgotten to buy a vignette, so avoided motorways until Lucerne when we could ask at Dilwyn's hotel. We arrived there in good time via the back roads, and proceeded to help ourselves to coffee

and biscuits in the unattended reception. I also reset the home page on their free internel computer in the lobby to the QL-Mac show. I also made a copy of a Luzern map on their scanner/printer. The receptionist arrived just as I was doing this, and I thought her odd look was because I was using her printer. No such thing - she thought I was the engineer she had called to repair it!

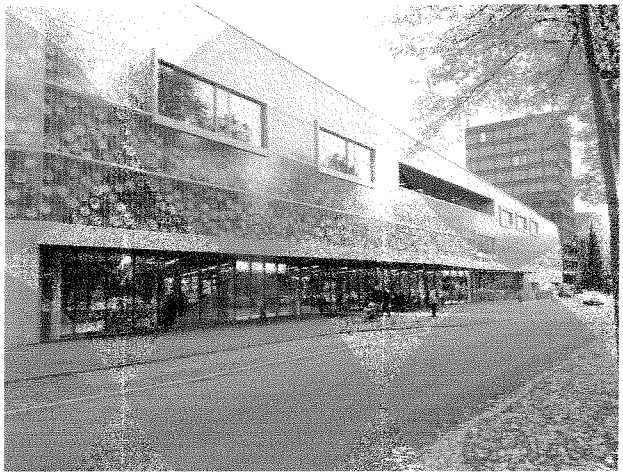
Off then for Geneva driving at first in the wrong direction. Satnavs, or at least mine, has no idea of direction until one moves. Why don't they all incorporate compasses? We had our first tourist view of the lake and Luzern - a really magnificent sight even (or maybe especially) through the mist. "Look there is the transport museum" said Simon, so we knew where the show was. They had steam trains and the like in full view behind full height glass windows. Why oh why do none of the London museums (especially the science museum) have such a display. You really don't need to say what it is, and it sells itself perfectly. A few seconds later, Simon said "There is Villa Maria". As advised by Dilwyn's hotel, we stopped at a garage to buy a motorway vignette. As I was sticking it to the windscreen, both of us noticed there was one there already! I expect the **first** tourist hirer in January buys one and they leave it there for the rest of the year, thus defeating the cunning government tourist trap. Amazingly the garage refunded me, so our budget (sorry about the car hire pun, and ours **was** Budget) was £20 better off.

Onward to Geneva and a double flash at 110kph in 100kph roadworks - my fingers are still crossed! The Satnav took us unerringly straight to Ian's door "....on the left". No matter it was on the right! I think the software assumed we were still in the UK as it did the same thing at Dilwyn's hotel. We *should* have chosen a German voice and changed it to French in Geneva - maybe they would have got it right. It was good to see Ian Pizer and his wife Eveline. I think the last time was at QL2000 in Portsmouth. He is quite frail now, but still pretty well, despite a balance problem.. I hope when I get to his age I will be anything like as healthy in mind and body. I then realised we had not brought any contribution for the stay. I needed to get petrol anyway, and managed to stop Eveline coming with me, although she then realised exactly what I was planning. They said it was impossible to find, and Eveline would look the other way. However my GPS had the petrol station in its database. The return was the really hard bit, but again no problem. On reversing into his drive I couldn't see a very low wall. I thought I was close to the wheelie bins so drove **very** slowly, and touched the wall. It is amazing how invisible the white marks were when I painted them with a black marker pen! Why don't **all** cars have bumpers like my Volvo - so solid they once wrote off a snazzy BMW that drove into the back of it, and there was no mark of any sort on my car!

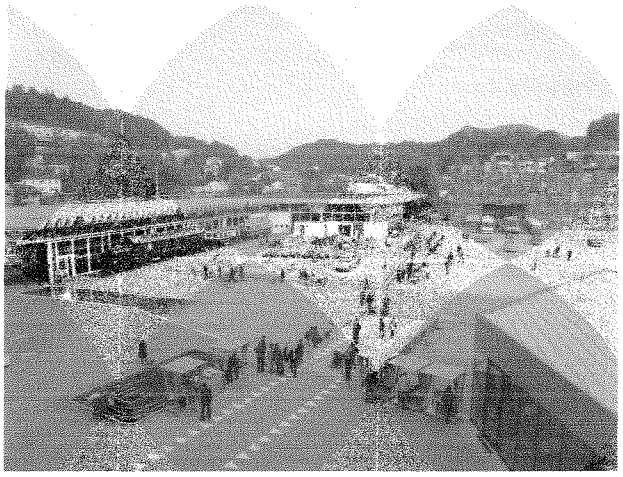
Ian had a new Iphone but could not get an internet connection using his wifi. I got a connection to the wifi router from my Macbook and an IP address, but no DNS. I correctly guessed the router IP address and the login - yes 'admin' and no password. DHCP was set OK, as the assignment of IP address suggested. It was set to the *latest protocol* - WEP! There was nothing else on offer, so it looked like the router was simply too old. However I found another very buried section, where it offered *Set up wireless*. This I followed and all the protocols were there, including the WPA2 I wanted. "What password do you want, Ian?". I will not expurgate the rest as it will spoil a later joke, but I doubt if anyone will want to drive to outside Ian's to use his wifi! "Chinese" he said. "No that is too simple - we need another word" so he chose *fondue*. Perfect we thought, there cannot be any such dish (see later). .... and of course the wifi then operated perfectly. It is quite odd that modern systems cannot now cope with WEP ***[Jochen cannot withstand to add two comments: first, all of the devices I know still understand WEP ... the Iphone is the only one I know which cannot handle it (another item on the list of things it can't do, which you take for granted) ... but maybe Apple adds it later, sells it as a great feature - who knows, as they keep doing this with other "features" which are standard on other phones for years. As for the WPA2 password - here I'm puzzled as I remember that my routers forced me to use at least 8 characters, and "fondue" is definitely shorter].***

The drive back to Villa Maria was totally uneventful and we took the last parking place. As we suspected from their website, it was quite OK, had free wifi (which worked) and a good view of the lake.

... and so to the show. I expect others will properly describe it so I will say little. We missed Urs' hour introduction, so he gave a quick 5 minute summary just for us. There was a very snazzy overhead computer screen projector, and Marcel even networked a machine to show demos in a Window (How did you do that Marcel? - I meant to ask). There were very interesting history lessons on Apple and DOS/Windows and a preview of Windows 7. I am not sure how that crept in but it was interesting. Its new search is a pretty straight copy of the Apple Spotlight, including the icon!

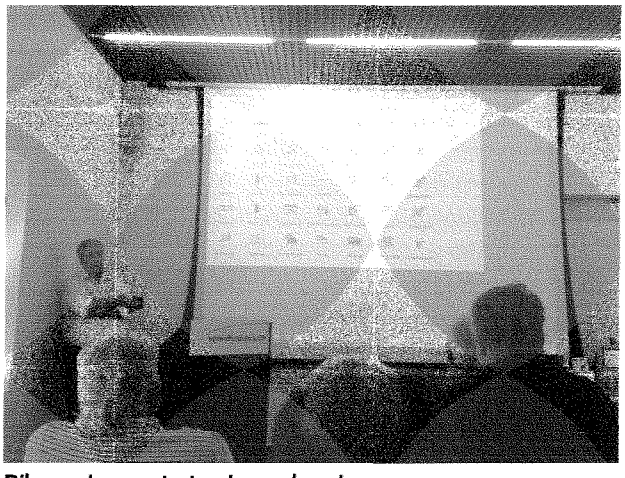


Verkehrshaus Luzern from outside



View at the exhibition (trains to the left, traffic signs right)

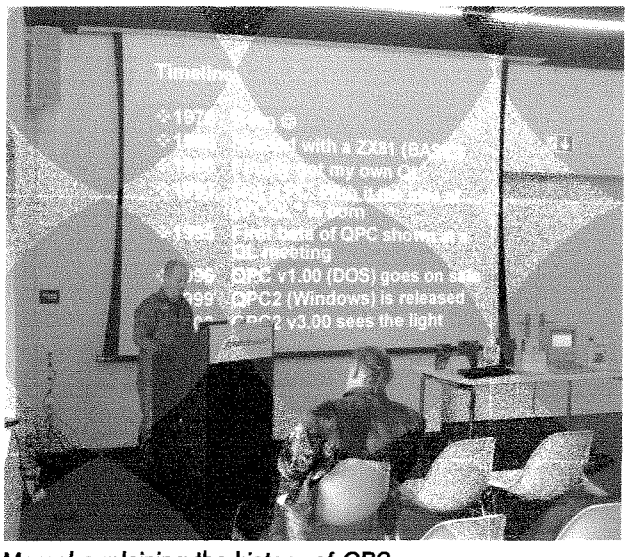
Dilwyn described his *Launchpad*, and I think we were all impressed that his very good icon-rich windows were all coded by him. The view of the lake and Lucerne was magnificent, but Urs often had to operate the motorised shutters to allow the screen to be seen. Adjusting the room lights though was a mighty hard job. Marcel closed the show with a talk on his life with QPC. I didn't realise he wrote the first version when he was 14. It was only a partly working beta, but people *demanded* to buy it. He didn't have a good classical education, and called the next version 'alpha'. This first encounter with the buying public forced him to bring it into production. It shows the power of the user. Without that encouragement, maybe we would never have seen it brought to life. Marcel's attitude toward the project is perfectly demonstrated by his introduction of TCP/IP - "just for fun". It was pleasing to see a thread in ql-users mailing list where someone had missed his earlier email and wanted to try it. A few simple lines of basic downloads a web page. Now who is going to write the QL browser? (8-)#



Dilwyn demonstrates Launchpad

Anton Preinsack from Vienna explaining the AMIGA history. He seems to be infected by the "QL virus" now...

Simon and I escaped on Saturday to play in the transport museum - thanks Urs for the half price tickets. It was interesting to see the technology from a Swiss viewpoint. Anyone would have imagined from the early flight exhibit that the Piccard brothers were the leading lights. There was less mention of the Wright brothers! There were plenty of Swiss planes and space modules, but no sight of the Swiss navy. We unfortunately decided to leave the trains for Sunday, but didn't manage to see them. Simon and I managed to crash a lie-upon early plane mock-up (Wright brothers), but I



Marcel explaining the history of QPC

Many original Sinclair items on display, here QL stuff... Of course, there were Macs, other Sinclair products (like the ZX 81 and ZX Spectrum in working condition, but also computers like the Thor)

think we were expected to, and it was only a computer demonstration, although pretty realistic. Outside there was a brilliant giant mosaic of old motorway signs. .... so Zug is near Lucerne. Heather, an alto in my church choir had moved there a few years ago, but I didn't know her address.

On Saturday evening we all met in a restaurant for a meal. This was our first wallet-opening view of the real Swiss world, and I was struggling to find anything remotely affordable. Up to then we had not had to spend any real money. The **only** thing cheaper in Switzerland was petrol. There was only one chicken dish - wings in a red sweet and sour sauce. I hate sweet and sour sauce. I couldn't afford a steak at about £35. I settled for what was clearly a tourist pauper's menu (as did everyone around me). I had two tiny meat burgers tasting only of Maggi sauce, some boiled rice, and, yes, a sweet red sauce. That cost about £12.50. Dilwyn told us about the hot water they gave him and Ann with some uncooked meat and vegetables in a hilltop restaurant. It cost over £60. They called it, wait for it, a *Chinese Fondue* (see earlier). Blow.

Simon and I walked back to Villa Maria leaving Marcel (our third man in the room) at the restaurant. I said I would text where I had put the outside door key ("Under rh shrub pot" - referring to my phone's *sent* folder), We though completely missed Villa Maria first time round. "Seems a long way". "Isn't that the transport museum?"

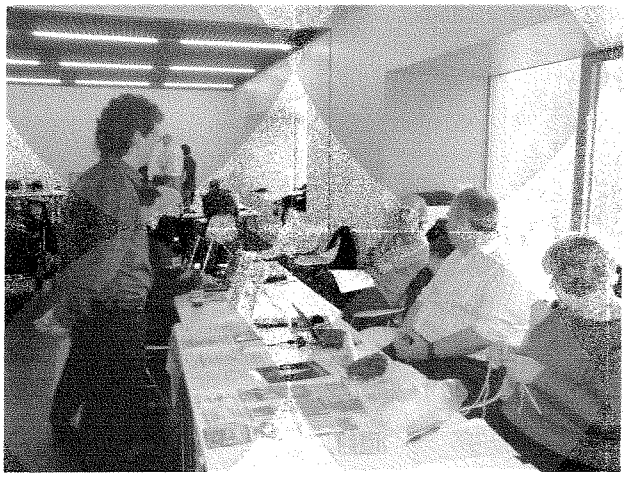
The show finished at 5:30 and Simon and I set off for Zurich with our brilliant souvenir rucksacks, with a show emblem made and sewn on by Urs - thank you Urs. First thing though was to get a good car wash. The car hire had already said they could not sign off a dirty car, and I had the scrapes. On the way we went *over* Zug (**Hallo young Heather wherever you are** - but that is *King and I* not *Sound of Music* - apologies to Oscar Hammerstein).and arrived in Zurich. "Lets not go straight to the airport but look for a nice local restaurant". We first filled with petrol, and then found ourselves right in the middle of a commercial area. Not a sign of food. After maybe 30 minutes we found a street absolutely stuffed full of restaurants including a VEGETARIAN one especially for Simon. Yippee. I parked but then realised we were too tight for time, so off to the airport. Budget signed off our sparkling waxed clean car and we had a pretty good and not too expensive pizza each, We then had what seemed plenty of time to go through security. No such luck, as it was a 20 minute journey including a train. I also had too complicated a bag for them and they had to separate **everything** and I was taken away to a dark room to be frisked. I thought maybe they were going to strip search me! I arrived at the Easyjet gate a good 10 minutes after it closed, but there was a long queue. Phew.

We made it back on time, it was dry, and the motorcycle was there and happy. Simon though had mislaid his wallet, and needed petrol. He had only cash, but maybe the Aylesbury stations were automatic only at 22:30 on a Sunday. I filled up his car from my spare petrol can, and he seems to have made it home.

Well done to Urs for organising an excellent show in an excellent venue, and he and the local user group paid the bills for the venue.

We may all be meeting again next summer in Vienna.

What is Simon doing there and why is Marcel smiling...?



Does this remind you of something...?

A tricky Trap

by George Gwilt

One of the most useful entries to the QDOS or SMSQ/E operating system for an assembler programmer is IO_EDLIN. This is a Trap #3 call with D0 = 4 which allows a line of characters to be edited. However, Andrew Pennel, in his guide to the QL operating system, The Sinclair QDOS Companion, says:

"This is potentially a very useful trap, but is tricky to use."

I have used this trap in several programs but I had a hard time getting it to work so I agree with Pennel. While adding items to Norman Dunbar's useful QDOSMSQ Wiki, described in QL Today Vol 13 Issue 4, I looked more carefully at IO_EDLIN so that I could describe it properly, and realised that although my programs which used it worked, it was more by good luck than good management. It was clear that I had not fully understand the trap's working.

Since others may also find the trap tricky I thought it might be useful to set out what I have found.

The definition of any piece of code in an operating system serves two purposes. First it shows the system programmer what must be coded. Second it shows an application programmer how to use it. These are two different purposes and in the particular case of IO_EDLIN I think the definition as it appears in all the places I have looked in has suffered as a result.

System Programmer's View

What IO_EDLIN must do is to arrange for a string of characters to be edited by a user pressing keys. The editing must allow the addition and deletion of characters, the positioning of a pointer to any character in the line and, finally, to the acceptance of the string. How should this be done? The string must exist somewhere. Hence the need for a buffer in RAM. Also the changes to the string must be mirrored on the screen. Thus we need a CON channel to be opened and we need a pointer to the buffer.

We can now see more clearly what the system programmer must do. He must read in one character at a time from the CON channel and act accordingly. A left or right arrow key must move a pointer to the current character in the string to the left or to the right with the proviso that the pointer must have a value between 0 and n, where n is the current length of the string. The value 0 will mean that the pointer is on the first character and the value n will mean that the pointer is one position to the right of the end of the string.

When a character is erased, all the characters to its right must be moved one position to the left. When a character is added the all the characters from that position to the end of the string have to be moved one character to the right. All this must be mirrored on the screen. One implication of all this is that the position of the cursor on the screen must be at the same character as the pointer in the buffer. Thus when a set of characters is moved in the buffer those characters must be printed to the screen at the current cursor position. When that has been done the cursor must be returned to that position.

There is one other value that the system programmer needs. That is the size of the buffer, otherwise it would be possible to input an unlimited number of characters with potentially disastrous results. When one of the characters, ENTER, LEFT or RIGHT is typed, the end of the string is signalled. For SMSQ/E the character ESC is allowed too as a terminator. The terminating character will form part of the string. This means that the length of the string itself must not exceed the length of the buffer less one. Attempts to make the string longer must cause the routine to exit with a buffer overflow error.

Having seen what the system programmer has to do, we can see what he must demand as initial parameters. These are:

- The position of the buffer A1.L points to the END of the string.
- The length of the buffer D2.W is the buffer's size (in bytes).
- The current length of the string D1.W is the string length (n)
- The current pointer in the string D1.TOP (a value from 0 to n)
- The ID of the CON channel A0.L is the ID
- D1.TOP is the high word of D1

From this the system programmer can see that start of the buffer is at address A1.L - D1.W. The end of the buffer is A1.L - D1.W + D2.W.

The pointer to the character in the buffer is given in D1.TOP

The system programmer will assume that the cursor on the screen is positioned on the same character as in the buffer. It is up to the application programmer to see that this is so. Help to the application programmer is provided here because when it starts IO_EDLIN prints the end part of the line from the position given in D1.TOP. IO_EDLIN then resets the cursor to its original position. You will recall that that operation is one that is needed inside the trap.

Application Programmer's View

Let us suppose that we want the string "ram1_prog" to be edited by:

```
Please edit: ram1_prog
```

First the characters "ram1_prog" must be put in a buffer of length 40 bytes say.

Second the characters "Please edit: " are printed to a CON channel whose ID will be put in A0. At this stage the cursor position in the CON channel will have x-position 13. However, it is now at the place where the string "ram1_prog" will be printed. The position to be set in D1.TOP would be 0 to signal the start of the string, not 13. If IO_EDLIN is activated with this information, "ram1_prog" will be printed to the screen and the cursor will be at the start of that string. By moving the arrow keys the cursor on the screen is now movable from the beginning to just after the end of the string.

It is also possible for the application programmer to arrange to print out the whole of the string himself instead of allowing IO_EDLIN to do it in which case D1.TOP must be set to 9 (the length of the string). When IO_EDLIN is activated now, it will print no characters and the cursor will remain at the end of the string.

If the application programmer wants the position of the cursor on the screen to be on the "p" of "prog" he will have to print "ram1_" on the CON channel and set D1.TOP to 5. This will cause IO_EDLIN to print the remaining characters and set the cursor on the screen to the character "p".

Trapping Overflow

Buffer overflow can only occur when the current length of the string is equal to one less than the buffer length. In that case overflow will actually occur if a character other than a terminator is typed.

If the application programmer does not arrange to trap buffer overflow then the results can be unpredictable. I would suggest that the simplest thing to do when buffer overflow is detected is to reset the string to just before the extra character was typed and re-enter IO_EDLIN. This can be done by printing the whole of the string (of length D2.W - 1) to the screen and by setting D1.TOP and D1.W to D2.W - 1. In this way IO_EDLIN carries on with the cursor being just after the end of the string. It would be useful too to emit a sound to indicate to the operator that something was wrongly typed but that is up to the programmer. At any rate the editing can now continue.

Final Comments

Pennel states that "on return there is no way of finding out where the cursor was when the edit terminated". On the other hand Opie, in QL ASSEMBLY LANGUAGE PROGRAMMING, says that D1 will have been updated. That is certainly true with SMSQ/E.

In fact, during the operation of IO_EDLIN, after each keypress the contents of both D1 and A1 are updated. For a left or right arrow key the value of D1.TOP is decreased or increased by one. For a deletion left, D1.TOP, D1.W and A1 are all reduced by one. For a deletion of the character under the cursor, D1.TOP is unaltered. If a character is added, all three of D1.TOP, D1.W and A1 are increased by one.

If a terminating character is pressed, it is added to the end of the string, not to the place where D1.TOP currently points. The value of D1.TOP remains unaltered, but the values of D1.W and A1 are both increased by one.

Thus the final position of the pointer to the character in the string can indeed be found from the returned parameters. It is in D1.TOP. The position of the cursor on the screen is D1.TOP characters from the screen cursor position of the start of the string. This position in the window could be found anyway by using the trap SD_CHENQ.

I think the main reason for the trap's apparent trickiness is the confusion arising from the definition of the parameter in D1.TOP. Although "cursor position" may be clear enough to the system programmer coding the trap, it can be very confusing to the application programmer. The true x-value of the screen cursor position is only the same as the value in D1.TOP if the string is printed at the start of a line rather than somewhere in the middle of the line. It might have been better to define the parameter in D1.TOP as the "character pointer" and to add a note saying that the cursor on the screen should be set to be on the character to which the "character pointer" points.

25 Years - Part 1

by Tony Tebby

25 years have gone by since the launch of the QL.

More than one person has suggested that I should write a little something to celebrate the occasion. This seems to me a bit strange. I am well aware that the launch of the QL was not as disastrous as the sinking of the Titanic or the 2004 Indian Ocean Tsunami. But celebrate it? Strange!

So rather than describing the development of Domesdos (the earlier name for QDOS), I thought I would look back at all the progress that has been made in system software, and, in particular personal computer systems over the past 25 years.

To appreciate the progress made, it is necessary to wind back the clock to 1983, the year that the ZX83 was not launched. Much has already been written about this rudderless project that started out as a development of a portable version of the Spectrum (ZX82) and ended up as a quantum leap into the void, so I will not repeat, confirm or deny it here. I will just try to give the background.

Then I shall describe where Domesdos went after the ill timed launch of the QL which was just a black shadow of the planned ZX83.

Finally, I shall give my own view of the progress made in workstation operating systems since 1984.

In the beginning

The QL started as the ZX83, a development of the ZX82, The impetus to create a new operating system for it came from the decision to replace the trusty Z80A microprocessor by a cut down MC68000. This "sexy" 32 bit microprocessor obviously needed something more impressive than the old spectrum software and Sir Clive decreed that it should have a "version of Unix that works". Later on, other requirements were added.

Depending on your point of view, the resulting system was a major breakthrough and outstanding success or it was a totally disastrous deviant.

For those who were not around at the time the idea of a "version of Unix that works" must seem bizarre. Surely, all versions of Unix worked, didn't they?

Unix in 1983

Well not quite, Unix "sort-of" worked provided you did not try to use it. At the time, UNICS (and later Unix) had been around for about 13 years, during which it had become legendary for its exceptional slowness and quirkiness. Unix had three great attractions for academics – it was free and so did not require a budget – it was portable and so could be implemented on new platforms in not much more time than it would take to re-write it from scratch, keeping thousands of students off the streets – it was quirky and so using it was a real challenge. The slowness was not an attraction, but neither was it a serious problem in the academic world.

Unix however had offered a glimpse of a different idea of an operating system. It was not so much what it did, but what it might have been able to do in a world with unlimited computing power, and if it had not been Unix. The problems were that computing power was not, is not and is not likely to become unlimited, and Unix was Unix.

The ill-fated XENIX system should give an idea of the slowness of Unix. When this was launched (just after the QL) on an IBM PC XT platform (about 4 times faster than a QL with 10 times as much working memory) one journalist coined the phase "a brain dead version of Unix" – an epithet that stuck. But it was not "a brain dead version of Unix", it was a real Unix running on a desktop computer which lacked the power of a $100,000 VAX (the favourite Unix platform of the period). A 2009 personal computer is several thousand times faster, with several hundred times more memory, than a early 1980s VAX, so, although Unix is still chronically slow, this is not now as obvious.

The slowness of Unix was, however, not the only problem: it also suffered from an operating system interface that was, as the French would put it, *bordélique* and it had a well deserved reputation for chronic instability.

A Unixly chaotic operating system interface

The chaotic operating system interface was a direct result of a design choice made by the developers – minimisation of the number of operating system functions.

This minimisation of the number of operating system functions had three effects.

The initial effect was that the real world had to be twisted to fit into the Unix minimalist world. For example, the I/O system was based on the paper tape reader / punch model: when console I/O and a data storage system were added, these had to be twisted to fit the paper tape model. While this had the advantage of facilitating the implementation of scripts (which are still fundamental to the operation of a Unix system), it had very serious consequences both for the primary use of the filing system (data storage and retrieval) and the interactive use of the console. So the whole I/O system was all turned onto its head and everything was treated as a file, even if it was an interactive device. This made it even more irrational.

The second effect was that anything other than the most basic functions had to be added on. This was great, everyone could have their own flavour of extended Unix – hundreds of different, incompatible commercial and university versions and thousands of private versions. Is this really a good idea? The developers and the academic establishment thought so – natural selection of the best version was obviously better that allowing a development group to impose an arbitrary choice. I did not find this such an obviously good idea.

The third effect was that functions that could be implemented simply and efficiently by a single operating system call were pushed out into C libraries where complex and inefficient routines had to make large numbers of "primitive" OS calls for each higher level function. Later this complex and inefficient approach was elevated to a virtue. Minimalist operating systems interfaces became "a good thing".

A Unixly unstable operating system

The instability of Unix was another concern. Just before the QL came into the world, Sun Microsystems was set up to exploit the nascent interest in Unix by making workstations using the most powerful microprocessors then available (3M machines: 1 MIP, 1 Megabyte, 1 Megapixel). These were sufficiently powerful for their SunOS version of Unix to seem comatose rather than brain dead. A great innovation in SunOS was the fast boot process to recover from system crashes quickly. Paradoxically, as one reviewer pointed out at the time, this made SunOS seem even less reliable than standard Unix: an ordinary version of Unix taking five minutes to boot could only crash 12 times an hour, whereas SunOS booting in less than a minute could crash 60 times an hour. Even when Unix was not crashing, the reliability, in terms having a system that did what you wanted to do rather than what it wanted to do, was not particularly good – to quote Dave Mankins[1] who slightly misquoted Johnson: "making Unix run securely means forcing it to do unnatural acts. It's like the dancing dog at a circus, but not as funny – especially when it is your files that are being eaten by the dog", evoking the propensity of Unix for destroying your data even in the absence of deliberate attacks.

Sir Clive's "Unix that works"

Sir Clive's idea of a "version of Unix that works" was, therefore, really rather revolutionary.

I interpreted "that works" as meaning that it should be efficient, reliable and with a rational operating system interface to remove the three main Unix problems. Some people have always objected that a "version of Unix that works" should have been efficient, reliable and with a standard Unix operating system interface. However, I took the view that the standard Unix operating system interface was not only a serious problem in itself, but also a barrier to making the system efficient and reliable.

There never was even an outline specification for the new operating system for the ZX83, Sir Clive did not work that way – he had an amazing capacity for delegation, for letting his "chosen" to get on with the job and for accepting the consequences himself if it all went wrong. So what were to be the salient points of a new operating system for the ZX83?

---

1 Unix Hater's Handbook http://www.simson.net/ref/ugh.pdf

Single user with pre-emptive time-sharing between tasks.

Unix was (and still is) a multi-user system. Although it was possible for a single user to pretend that he was several users, this provided only limited multitasking with negligible interaction between tasks: "native" pre-emptive multitasking allowing operations on shared data structures was not to become available for many years.

At a distance, it is difficult to imagine why pre-emptive time-sharing between tasks was thought to be a good idea, but there was a group at Sinclair that was working on parallel systems, and they thought it would be useful to be able to have a hundred programs running simultaneously, sharing memory, for simulating a massively parallel processing system. So one hundred application programs running simultaneously became the target.

Maybe it was not such a bad idea. After all, this was something that you could not do using a $10,000 Apple Lisa, a $20,000 Sun workstation or a $100,000 VAX[2] and the typical Sinclair customer was an enthusiast.

From a more mundane point of view, this did allow little features such as clocks to be implemented without requiring the "main application" to continuously call a function to pass control to another task. It did allow background communications tasks to receive and transmit data without the "main application" being affected. But, in itself, it could not directly handle "switching" between applications. Why not?

Because, if you had two or more applications running simultaneously they could both be writing to the display at the same time: which would you see? If a user typed something, which application would get the keystrokes? There were two reasonable solutions to this problem: having only one application "active" at a time (single task switching) or windowing – it being much easier to do both as in the early Apple Mac.

Monospaced text I/O on bit mapped display.

This was to become, possibly, the most disappointing feature of the QL software. There were reasons for this being the obvious choice.

1. The display handling was all carried out by a device driver that could be (and was, many times) replaced, even while the machine was running. A graphical user interface (GUI), was, therefore, not a baseline requirement.
2. The conventional schemas for implementing these GUIs all placed a very heavy interface burden on applications, making it very difficult to write software for the machine. Moreover, Sinclair had contracted Psion to port an office suite designed for the monospaced text PCDOS environment.
3. Existing systems with GUIs were, to say the least, extremely limited and slow even with hardware much more powerful than the QL.
4. The Sinclair computer range had, from the start been a "instant programming machine". This is totally anti-GUI.
5. Time.

Device independent filing system.

Not the Unix "everything is a file" syndrome, but allowing, for example, data to be read from a storage device as if it were coming from a console (a line at a time) and files to be read or written (handling file length, properties and end of file) over a communications port as if you were accessing a storage device, while providing a clear separation of distinct input and output functions.

Real-time hardware management.

The QL hardware was designed using well established Sinclair principles: do not do it in hardware if you can do it in software. There was no question of trading off cost against performance. The software had to respond to a general interrupt, identify the interrupt source and transfer information to or from a device driver or application in a handful of instructions – unlike any other PC or workstation, there was no hardware buffering, FIFO or DMA.

2 Five years later, contemporary reports on the spread of the Morris Worm estimated that 1988 VAXs running BSD Unix were unable to handle more than 20 active processes.

25 Years - The state of the art c.1983

Like most computer users and software developers at the time I had suffered from conventional operating systems I could understand the potential of Unix but I did not understand why it was so exceptionally bad. It was supposed to be simple, which should make it efficient, but its slowness was already legendary. It was supposed to be simple, which should have made it reliable, but it wasn't.

So, there I was, I had a few months to turn Unix into a working system – a task that had defeated thousands of developers who had been working on it for years – or had it?

The answer did not come in a flash, but when I finally got there, the answer was simple. The Unix that we knew had not been designed to work and the developers had not been trying to make it work. This Unix was the result of the application of academic theories of no relevance to the real world.

The Why is easier to explain than the How.

Why was Unix so bad?

That is very simple. Unix came from the academic world. Nobody's job was on the line. Whether it worked or not ceased to be an issue after the first version printed "Hello world". What was important was the application of "modern operating systems" theories. At the time, the operating systems provided by the major computer manufacturers had no theoretical basis – they were just cobbled together to meet ad hoc requirements. On the other hand, the developers of these systems did not have cosy jobs for life in research, so that if these systems did not work, their developers would soon be looking for new jobs. This was a fairly powerful incentive in the days before employment protection.

As a scientist, (I have a certificate to say that I have a degree in physics, for what it's worth) a sound theoretical basis is something that I wholeheartedly welcome, but, being a pragmatist, I feel that having a working system is far more important.

How did Unix get to be so bad?

What intrigued me in 1983, while the Unix story was still unfolding, was that the main platform for Unix was the VAX series of computers. It was all too obvious that the operating system running on these VAXes just could not be the same as the UNICS that printed out "Hello world" from a PDP7 in 1970 – The PDP7 was about as powerful as a ZX80. There was clearly something *louche* in this story that no-one was owning up to The Unix we knew could never have even given an indication of working on a PDP7. I managed to get hold of some documents about the development of Unix, written by Dennis Richie, amongst others, which shed some light on this anomaly. Although this is not the way it was put in those documents, it seems that Unix had been the victim of the upsurge in computer science that occurred over a very short period from 1962 to 1965. At that time, the leading lights in computer science were unanimous in their view of the future of computing: the near future (the 1970s) of computing was enormous time-sharing systems serving thousands of users. There was no possibility of building a single computer powerful enough to serve thousands of users – the natural consequence was that the future was in "symmetric multiprocessing" with massive arrays or networks of thousands of processors working in parallel.

The two problems to be addressed in this symmetric multiprocessing future were managing the conflicts between different processors for shared resources and distributing the users' workloads between the processors. The solutions to both problems had to be transparent and equitable (for more details, see "Modern Operating Systems", 1990 by Prof A. Tanenbaum, who still believed in these theories 25 years later).

The multiprocessing model

There was a unified model of software execution (the "multiprocessing" model) underlying the theories that were developed to meet this challenge. I am not sure that this was ever explicitly defined, I think it was just a private consensus. This model treated all hardware configurations, single processor computers, multiple processor machines with shared memory, loosely coupled networks of computer and any other imaginable configuration as being equivalent, presenting the same problems and accepting the same solutions. Above all, it elevated "symmetry" to the status of an inviolable law.

This was an approach that is very attractive: it provided solutions that were generalised and it promised a sound, universally applicable theoretical basis for operating system development.

Unfortunately, the real world was and is rather different. Problems of sharing memory that can occur on tightly coupled processor systems cannot occur on a distributed system with independent memory for each processor. Likewise, solutions that make use of shared memory cannot be used if the processors do not share memory.

I will not bore you with the almost endless list of real differences between different real hardware configurations that mean that, even where common problems can be identified, common solutions will be at best suboptimal and frequently totally unworkable on real systems. As a result, single processor multitasking systems were considered to be just a poor emulation of multiprocessor systems and so were required to emulate all the problems of these multiprocessor systems, even though these problems were not intrinsic to single processor systems.

The scenario was surreal. It was as if an eminent group of transport engineers had set out the design rules for future transport. This would mean that cars, lorries and trains could not have wheels, because they are useless on water. Ships could not have underwater propellers because roads are solid. Aircraft would have to fly at ground level, because a ship or train taking off could be quite dangerous. The generalised solution is, of course a hovercraft – a surface (land or sea) following airborne craft. In future, all transport would by by hovercraft. This metaphor is not an exaggeration, quite the opposite. A hovercraft at least has some useful applications, but I could not see any evidence that this mass of 1960s computer science theories could ever be applicable to any real system.

The surreal bit was not the theories themselves, but that, instead of being laughed into oblivion, they were actually taken seriously and were still being taught as inviolable gospel truth to computer science students. (25 years later this is still happening!)

Symmetrical multiprocessing

The unshakeable, immutable, unconditional, belief in symmetry had three main sources.

The first was a naïve idea of fairness that arose from the only form of computing envisaged: multi user time sharing systems. Symmetry should provide total fairness, but in practice it does not. Even the aim is not very sensible: surely it is better that all tasks are completed quickly if unfairly rather than slowly and equitably.

The second was a naïve idea of simplicity. It was felt that it was simpler to have all processors being equal, and, therefore, all programmed the same way. It is difficult to imagine that anyone could be so naïve as to believe that it would be simpler to have many processors each deciding which tasks were to be executed and fighting over resources rather than having a dedicated controller. But they did.

The third was a simplistic idea of reliability. The model was that of an ant colony. Individual ants can be killed but the colony carries on regardless. The big fallacy is thinking that the loss of a processor does not matter – it does: the data being processed is lost and if it is your bank account that is lost, you would think it mattered. The controller in a asymmetrical system is usually presented as a weak point. It is not. A controller failure will not lose data and it can provide recovery from individual processor failure with a simplicity and reliability that would be unimaginable in a symmetric system.

MULTICS and UNICS

The first major attempt to put these symmetric multiprocessing theories into action was in the development of the multi-user MULTICS operating system. The story of the MULTICS development project and the subsequent attempt by two of the MULTICS development team to salvage their *amour propre* by developing their own system, UNICS, on the side, is now the stuff of legends. A thoroughly revised anodyne history of the creation of Unix can be found in Wikipedia while another view can be found in "Multicians strike back"[3].

But the story I found in the "original sources" was rather different. Quite a lot of passages in these original sources were clearly untrue. There was a categorical statement that "fork" was the only

3 http://www.multicians.org/myths.html

MULTICS feature incorporated in UNICS. MULTICS was designed as a set of concentric shells round a kernel, UNICS was designed as a shell round a kernel (notice the similarity). MULTICS used a virtual machine execution model and, as "fork" works by replicating a virtual machine, UNICS also used a virtual machine model (coincidence? I do not think so) and so on. MULTICS had an execution model based on processes, UNICS had one process and later versions multiple processes (spot the difference).

I do not think that Richie et al were trying to mislead us, the striking similarities were probably more a result of UNICS being designed using the same dogmas as MULTICS, by people who had worked on MULTICS.

Despite this there were many differences (UNICS had, for example, separate "process space" and "files" unlike the MULTICS combined process and file space), but, in their various descriptions, the authors pointed out only one deliberate divergence from MULTICS: they abandoned the symmetric multiprocessing of MULTICS with its associated problems of "competition for resources" and the 1960s theories for dealing with these problems (including the synchronisation / mutual exclusion horrors). I could find no suggestion anywhere in the documents that mutual exclusion was omitted because the theories were fundamentally flawed: the authors apparently regretted leaving it out – they did so only to simplify the system so that it could be made to work.

I repeat "simplify the system so that it could be made to work". That was the key, UNICS was designed to work, it was not designed on the basis of academic theories.

Furthermore, when UNICS was extended to handle more than one process at a time, mutual exclusion was not re-introduced into the kernel. The authors noted that UNICS worked **despite** the omission of mutual exclusion, but they do not appear to have considered the possibility that UNICS worked **because of** the omission of mutual exclusion.

UNICS and Unix

Over the next few years UNICS became Unix and as it was ported to more and more powerful computers, developers started to put back into Unix those things that had been left out of UNICS – no wonder that it hardly worked any more.

By 1983, Unix had appeared on a small number of "executive workstations" (or rather executive toys) such as the Three Rivers Perq and the Sun, although, at that time, "the industry" treated Unix as a joke. Most computer manufacturers thought that their customers would prefer to have their payroll output reliably and correctly every Friday rather than sit typing commands such as `grep "\<[tT]he\> end"` all day. Or, as I heard it "How can you trust an operating system whose commands sound like body functions" (ps, sh, fc, grep, awk). The main usage of computers at the time was carrying out the same operations every day, week, month or year, reliably and predictably – in other words, boringly. Unix, on the other hand, could do almost anything – and to make it even less boring, it did do almost anything, regardless of what you might want to do.

25 Years - Out of the morass . . .

The starting point for Domesdos

Basic design criteria and philosophy

There were 5 basic design criteria for Domesdos

Compactness

Unlike most executive toy and personal computer operating systems, the self contained operating system for the QL had to be resident in a target 16k ROM.

Efficiency

It might seem obvious, but as the raw power of the QL was less than that of the first 1981 IBM PC, the operating system needed to be efficient.

Reliability

This might seem rather odd for a company like Sinclair which did not have an outstanding reputation for the reliability of its products, but there were two reasons for this, although both would disappear before the first machine was delivered. The first was that the operating system was to be delivered in ROM and it was not easily upgradeable. The second was that the machine was targeted at a more "professional" market than earlier machines because Sir Clive did not want to be in the games market – he wanted to be taken seriously.

Predictability

The dominant form of "serious" on-line computing was connection to a multi-user timesharing mainframe. This form of office working had created a new stress syndrome. A major contributing factor was the annoyance or frustration caused by highly unpredictable response which varied from sub-second to tens of seconds. The predictability of the response to user actions had become major requirement.

Accessibility

The general philosophy of a multi-user system (and this includes Unix) is of a **restricting** system whose primary aim is to restrict users access to prevent them taking control of the whole system. Domesdos, however, was to be an **enabling** system to maximise the accessibility of the system and hardware functions for both specialists and hobbyists.

Things to avoid in Domesdos

A good starting point seemed to be defining **THINGS TO AVOID** (the capitals are for fans of Terry Pratchett's Discworld – to be spoken with a hollow, death-like voice). These things to avoid were those academically popular ideas that seemed to lead straight towards complexity, poor or unpredictable performance, fragility or any combination of these.

1. Wilful ignorance
2. C programming language
3. Object oriented programming
4. Virtual memory / virtual machines
5. User based security
6. Synchronisation
7. Minimalisation

These **THINGS TO AVOID** are described in more detail in Box 1 and the means used to avoid them in Box 2. I did not realise at the time that these things to avoid would become the objects of worship by a narcissistic idolatry cult popularly known as the "Computer Scientists". I suppose that I should now call them the "Seven Cardinal Sins of System Design".

Good intentions . . .

The grand plans for a super operating system were derailed by a whole series of compromises required to fit Domesdos into the world of the QL hardware, to support the Psion office suite written for a CGA display on an MS DOS based IBM PC and to accommodate rapidly evolving in specifications and target markets.

Box 1 – Domesdos's things to Avoid

The seven cardinal sins of operating system design as seen from 1983.

1 Wilful ignorance

It should seem obvious that if you wish to build a system that works well and reliably, it is a good idea to know its performance and know its limits rather than sticking bits together following a set of arbitrary (possibly inappropriate) set of rules and hoping that it does the job. Apparently, this is not obvious.

While a certain amount of care is required to produce efficient code and there are some trade-offs between efficiency and code size, inefficiency comes mostly from not bothering to quantify the costs of operations and, therefore, wasting valuable resources through sheer laziness. Similarly, a system is likely to have a very unpredictable response if no effort is taken to evaluate worst case (or worst likely case) behaviour.

In all the documents concerning the development of Unix, I did not find a single document with calculations of the cost of any basic operating system function. The authors did not seek efficiency so Unix was inefficient by default.

In all the mass of 1960s theories on "multiprocessing" I did not find a single typical or worst case cost calculation to justify the complex mechanisms proposed for managing "competition for resources" or protecting "critical sections": these theories relied on asserting the "obvious superiority" of something that was not obviously superior.

2 C programming language

If you are going to program a version of Unix, C would be the obvious language, would it not?

I was not convinced. The various documents I had found about the original versions of Unix gave some very interesting figures on timescales. It appeared that the rewrite of Unix in C took less time than it took to write the original version in machine code, but not by much, whereas rewriting a piece of software should take much less time than writing from scratch (because you know exactly where you are going). Furthermore, the first time the C version of Unix was ported to another machine, it apparently took longer to adapt it than it had taken to write it for the first time. On the face of it C wasted time rather than saving it.

C represented a specific computer instruction set which was not appropriate and, even worse, it was tied to the Unix environment and concepts and, therefore, likely to induce typical Unix errors.

Finally it was so ill-conceived that, while it was possible to do really stupid things writing in machine code, writing in C you could do really stupid things without even knowing it.

3 Object oriented programming

Object oriented programming is based on "encapsulation" a fancy term for hiding all the dirty little tricks you do not wish others to know about inside a hard shell. Furthermore, rather than being explicit about the operations that are carried out, and how they are done, every operation is implicit, abstract or both: programmers are not supposed to know what goes in inside an object. The result is that nobody knows how a system created using object oriented programming works because nobody is supposed to know. It is all deep magic (when it works) or wilful ignorance (when it does not) – a **BAD IDEA**.

To cap it all, all operations using objects are stunningly inefficient. You need a byte of data from an object? It should take one machine instruction. With object oriented programming it takes at minimum tens of instructions and can be several hundred: just to make the system obscure.

4 Virtual memory and virtual machines

These two concepts are entirely independent but, as both require a dynamic address translation unit (a unit that converts the "virtual addresses" seen by an applications program into "real memory addresses"), they are often associated.

The supposed advantage of **virtual memory** was that it allowed the system to degrade more gently when there was a shortage of memory allowing systems to use less memory. This theoretical view is the result of a dramatic oversimplification of memory allocation processes. Experience pointed to the opposite conclusion. For example, in the late 70s when changing from IBM MVT (real memory system) to MVS (virtual memory system) the main memory had to be doubled in order to handle the same workload. This experience was repeated many times on many different systems.

The **virtual machine** model is a fundamental part of the 1960s dogma for multi-user systems. The idea is that each user "sees" a virtual computer that is completely isolated from the virtual computers seen by all other users, thus providing a naïvely simplistic security mechanism. This was, of course, totally irrelevant to personal computer usage where there is only one user and it had already be demonstrated to provide a fundamental security breach rather than a security mechanism. The other main drawback to the virtual machine model is that most operating system functions are concerned with transferring information to, from or between tasks - while the virtual machine model not only made systems more vulnerable, it made inter task communication both more complex and more costly.

5 User based security

In 1983, the dominant form of "serious" computing was time-sharing a multi-user central computer system. The same scenario formed the basis of the 1960s multiprocessing theories. For this type of system, security was limited to preventing individual users hijacking, using or corrupting other users' data. Oddly enough, UNICS, which was designed as a single user system, had user based security concepts from the start. For a personal computer (single user workstation) the multi-user problem does not exist, so there is no need for user based security mechanisms. At best they are merely obstructive and annoying while giving a false sense of security.

Unix had two separate user based security mechanisms: the "process" model of program execution and the file system owner/group/all and root/notroot concepts.

Processes are closely related to virtual machines. As Unix type virtual machines are more of a security risk than a security mechanism, the Unix process model merely makes a single user workstation more vulnerable.

The Unix owner/group/all concept of file system security was almost unworkable on multi-user systems as it assumed a strict hierarchy implying that each user belongs to only one group, and that only one group could be allowed access to a file. For a workstation it was totally ineffective: where a machine has accessible file store media (even if you need a crowbar to access it) or can be re-booted to a different operating system on a external drive, removable medium or over a network; the **only** effective mechanism against data theft is file encryption (using per-file keys NOT per-user keys) and there is no protection against data loss or destruction except for mirroring the data on remote storage.

6 Synchronisation

In the 1960s a whole edifice of theories was built up on the basis of using synchronisation as a means of providing mutual exclusion to resolve access conflicts between "processes". In fact this was misleading. The theories were not concerned with resolving the conflicts themselves, but concerned with resolving the problems arise when mutual exclusion is used to try to deal with these conflicts.

This created a self-sustaining spiral. The basic mutual exclusion theories simply made the underlying problems worse, which led to the development of synchronisation theories which exacerbated the problems of mutual exclusion which led to more theories...

In conventional systems, synchronisation mechanisms had also been adopted for signalling between tasks, for example indicating that data was available for processing. This too had proved to be the source of many fundamental system design problems.

Avoiding synchronisation and all its associated nasties was, therefore, a primary design aim.

7 Minimalisation

The concept of minimalisation is associated with the "less is more" and "worse is better" system design philosophies that developed in the 1970s and eighties to justify increasing idleness and incompetence.

The principle is that by minimalising the operating system functions, the complexity is pushed into the application programs, making it simpler and easier to design the operating system.

In practice the effect of this approach is rather different. As an operating system should not just be considered to be a set of core functions but the whole of the support for the applications programs, minimising the core functions has the effect of increasing the complexity and size of the "higher level" functions providing applications support.

It becomes even worse when the minimalisation is compromised. A real minimalist approach to reading data is to treat input from any device of any type as a stream and have just one "non-blocking" operating system call to read a either one byte or a given number of bytes from the stream. As the call returns immediately whether the read is complete or not, then this call can be used for both checking for input and reading from a file. To read any data that was not instantly available, the program would have to cycle in a tight loop retrying the call, which in most cases would be unnecessarily complex and inefficient.

The first typical minimalist compromise was to provide two calls: a (non-blocking) call to test whether there is data available and a (blocking) call to read a fixed number of bytes from the stream.

This did not solve the problems. It did not allow for keyboard input where the user may type characters and then edit them before hitting ENTER. This meant that the minimalist approach was then further compromised by introducing switches changing the behaviour of the read bytes function depending on the device and how the application wished to interact with it, increasing the system complexity significantly.

The end result is that compromised minimalisation not only makes applications programs and application support software inevitably more complex than providing an appropriate range of core functions, but the minimalised core functions themselves are very likely to be more complex than more complete set of regular core functions.

Box 2 – Avoiding the things to avoid 1

1 Avoiding wilful ignorance

All the Domesdos system data structures were completely defined (with provision for expansion) before any part of the system was coded.

In design, the execution time of all critical sections of code was calculated for both typical and extreme scenarios. For example, the scheduler design was fixed when it was able to schedule 100 application programs, active or waiting for I/O, with a worst case overhead of 50% of the processor time. The performance of the system was known before it was coded.

All timing critical services interfacing directly to the hardware had known worst case timings.

2 Avoiding C programming language

Conventionally, operating systems had usually been written in assembler (a family of programming languages based directly on the machine's instructions: one line of assembler translates into a single instruction). Unix was a notable exception.

Domesdos was not, however, written in machine code or assembler. It was written in pseudo code (the fancy name for any representation of a program using rules that are made up as you go along) which was then "hand compiled" even though hands had nothing to do with it.

This ensured that the implementation was not constrained by the limitation and peculiarities of C or any other programming language.

3 Avoiding object oriented programming

Any one with a knowledge of the principles of object oriented programming looking at the structure of Domesdos might think that the attempt to avoid object oriented programming had failed completely: every item in memory, including jobs, could be considered to be an instance of an object complete with constructor, destructor, and a variety of methods and properties.

A channel to a file, for example, could be considered to be an instance of a "file channel" object which added file specific methods and properties (position, flush, etc.) to the methods and properties (read, write, etc.) inherited from the "I/O channel" object which itself inherited basic methods and properties (create, destroy, ownership) from the "memory" object.

The Domesdos approach was, however, very different. Domesdos used "data design", a slightly earlier concept which, because of its simplicity, found little favour with academics. Most programming languages are algorithmic or procedural and not particularly concerned with data. Object oriented programming is the apogee of the procedural approach as the data is completely inaccessible.

Data design was a programming approach that took, as its basis, the primordial value of data and the relative insignificance of procedures and algorithms. This is not an ideal approach for calculating the value of PI or drawing fractals but, then, as now, most computing outside research laboratories was concerned with data handling rather than intensive calculation.

The principle of data design was that data structures should be designed to be well defined for all possible states. For example, rather than writing an algorithm or procedure for suspending a job, the executing and suspended states of the "job control data structures" are first defined and then the code for suspending a job "writes itself".

There are similarities between the Domesdos use of data design and some of the aims of object oriented programming. A "file channel block" had all the data structure of a basic "I/O channel block", so that all code that could operate on an "I/O channel block" could also operate on a "file channel block". Likewise, an "I/O channel block" had all the data structure for a "memory block", so that all code that could operate on an "memory block" could also operate on an "I/O channel block" and a "file channel block".

There are also major differences. Because the data blocks in Domesdos were defined explicitly, the Domesdos file system device driver (privileged code) could not only access the file channel block defining a "channel" from the application to a file, but also the associated filing system block (shared between all files open in a particular filing system), the associated physical device block (shared between all filing systems on a disk) the associated disk interface block (for all disks on a particular bus) and the operating system block which held all information common to applications, device drivers and hardware.

This simplicity led to ridiculous accusations that the system was unsafe: an error in the privileged device driver would not necessarily be contained. This is absolute nonsense based on the academic view that reliability is a matter of keeping the system going regardless of how much damage is being done to the data. An error in the filing system will destroy data: the system will be broken whether or not other system structures are damaged. Intrusive containment measures simply increase the complexity and, therefore, the increase the probability of there being errors while reducing the probability that those errors will be detected.

The data design principles used in Domesdos, therefore, provided the useful features of object oriented programming in an open, clear, explicit, efficient, natural way instead of the closed, obscure, implicit, inefficient, object oriented way.

4 Avoiding virtual memory and virtual machines

Avoiding virtual memory was not difficult as the hardware did had neither dynamic address translation nor fast backup storage. The memory management strategies used did not, however, preclude the use of virtual memory.

It would have been possible to implement a virtual machine memory model, by shuffling the contents of memory on every task switch, but as this would merely have added to the inefficiencies inherent in the virtual machine model, a real address memory model was implemented.

5 Avoiding user based security

The Domesdos application task model was based on the classic "job" concept. I have seen Domesdos jobs described as processes, but they certainly are not. If you were trying to be contentious, a Domesdos job could be described as combining all the advantages of Unix processes and Unix threads while avoiding the drawbacks of either. But I will not describe them that way as Unix enthusiasts are not noted for their sense of humour.

Although the *hardware* did not support any form of protection against accidental or deliberate corruption by one task of the data belonging to another, Domesdos did have a rights system in the form of "ownership" and "usership". This rights system could have been enforced if appropriate hardware had been available. The "ownership" and "usership" of data and program memory blocks made the system self-cleaning provided that tasks did not abuse the rights system.

A Domesdos job has its own code base and data space. It can spawn independent jobs having no access to the spawning job's code base and data space (like processes). It can spawn dependent jobs which, by virtue of the separation of "ownership" and "usership" rights, may have their own code base and data space (like processes but unlike threads) and may access their owner's code base or data space (unlike processes but like threads).

As user rights to files on a workstation are completely unenforceable, files were not flagged with user rights. Per file encryption, which would have been the only effective data protection mechanism, was considered too complex.

6 Avoiding synchronisation

The earliest versions of Unix did not use synchronisation mechanisms for dealing with access conflicts; they relied on operating system calls being atomic unless voluntarily suspended. For application programs, where a response time of some tens of milliseconds is adequate, this is an simple, efficient and safe approach and it was used to a certain extent in Domesdos.

It is, however, unsuitable for dealing with contention for access to shared data structures between interrupt servers and other software. Rather than using invasive mechanisms such as disabling interrupts to protect "critical sections" or, even worse, using symmetric synchronisation mechanisms, Domesdos implemented a range of asynchronous, asymmetric access mechanisms. For example, if an interrupt server needs to release a scheduled task, it can do it at any time, even while the scheduler is in the process of rescheduling, without any lost events or any precautions being required in the scheduler code or the interrupt code to prevent access conflicts. These mechanisms do not have any "critical sections" and so, in Domesdos terminology, they were called "intrinsically safe". These mechanisms were developed specifically for Domesdos but some of the ideas were partly based on the concepts for asynchronously updating distributed databases that were being developed by the systems group at the CADCentre, my previous employer.

Synchronisation mechanisms were also avoided when flagging completion of asynchronous processes such as transmitting or receiving data on an I/O port. Events (intrinsically safe) were used instead.

7 Avoiding minimalisation

Rather than seeking to minimise the operating systems interfaces, Domesdos sought to regularise the interface by providing a broad, coherent set of basic functions. Using a simplistic analogy, the broader the base, the more stable the edifice built on it.

The best example was the I/O sub-system. For reading data, separate calls were provided for testing, reading a single byte, reading a "line", reading a string of bytes and unbuffered direct reads. There was no arbitrary "blocking / non- blocking" behaviour on individual calls: all calls had a timeout parameter from 0 to 10 minutes (or wait forever) whether or not a timeout had any sense for a particular call.

Because of this regularity, and because the operating system itself handled the timeout, buffering and event signalling, writing a comprehensive IO device driver for Domesdos was much easier than writing a primitive IO device driver for Unix or even MSDOS (or at least it did seem that way to me – some accurate documentation would have been a help for others!).

QL-Aided Design - Part 2

by Simon Balderson

Here is part two of QL-Aided Design. In the previous article the values for the volume control network were found by the QL. This article deals with the resistor/capacitor network attached to two taps on the control to give tone compensation depending on the volume setting.

Figure 1 shows the circuit diagram of the control. This is taken from an article entitled "A Two-Tap Bass and Treble Compensated Volume Control" by William O. Brooks published in Audio Engineering, August 1951, pg. 15. VR1 is the chain of resistors whose values were found by the first program. This article deals with the program to calculate the capacitor values for C2 and C4. Their values need to be calculated for differing total resistances of VR1.

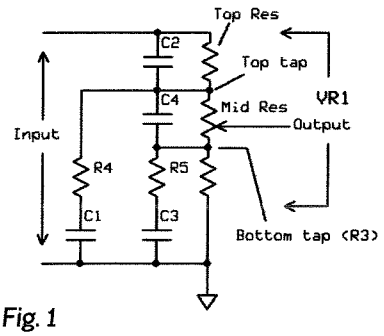


Fig. 1

The bass boost is given by C1, R4 and C3, R5. These remain fixed and do not need altering for different potentiometer resistances. The frequency around which C1 and C3 work is 400Hz (the turnover frequency). When the reactance (resistance to a.c current) of C1 equals the resistance of R4 the turnover frequency is 400Hz. Above 400Hz the reactance decreases thus pulling down the voltage at the junction of R4, C2 and C4 and attenuating the treble. Below 400Hz the reactance of C1 increases effectively boosting the bass. R4 helps to limit the effects of C1 which on its own would have too big an effect on treble frequencies. C3 and R5 perform the same bass boost function on the lower tap of the control.

The tapping points are fixed at one third and one sixth of VR1's total value. With differing values of VR1 the resistances between them will change. The turnover frequencies for treble compensation are 3500Hz for C2 and 5000Hz for C4.

A value for C2 needs to be found whose reactance is equal to that of TopRes (R1) when the frequency is 3500Hz and similar for C4 and MidRes (R2) when frequency is 5000Hz.

The formula for calculating the reactance of a capacitor is:

$$X_C = \frac{1}{2\pi f C}$$

We already know the frequency, Pi multiplied by two and the reactance value so if the formula is juggled then the QL can find the mystery value of capacitance for us.

$$C = \frac{1}{2\pi f X_C}$$

Here is the program itself. It has many similarities to the volume control program and again is written using SMSQ/E in High Colour mode running under QPC2. Figure 2 shows the background image loaded into channel 7. As with the previous article the image was created with Adobe Photoshop Elements. The two picture inserts were scanned from the original article in Audio Engineering. Dilwyn Jones's website provided the Helvetica fount: **www.dilwyn.me.uk/fonts/index.html**

My copy of the Helvetica fount was in a file called FONTVIEW_ZIP (which I downloaded a few years ago).

Lines 180 to 250 form a loop to input and check the resistance value of the potentiometer. Lines 390 to 560 calculate and print the values on screen. Note the use of OVER 1 to ensure that the text is printed on a transparent strip so as not to spoil the background image.

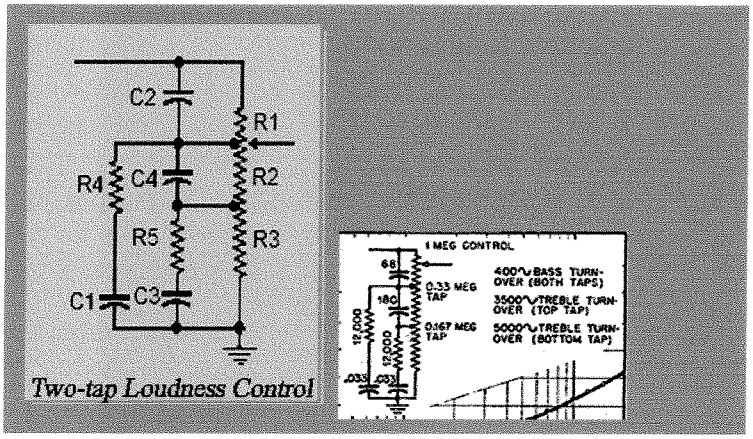
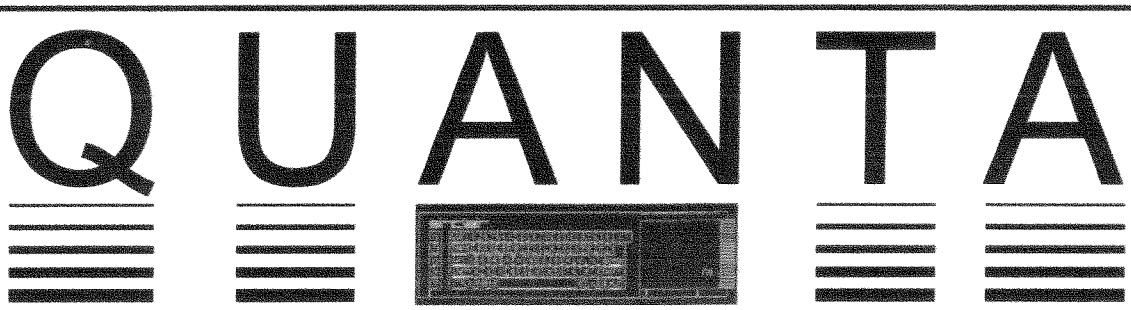


Fig. 2



Independent QL Users Group

World-wide Membership is by subscription only,
Offering the following benefits:
Bimonthly Magazine - up to 52 pages
Massive Software Library - All Free!
Free Helpline and Workshops
Regional Sub-Groups. One near you?
Advice on Software and Hardware problems
Subscription just £14 for Full Membership

PayPal (see QUANTA Web Site),
Cash, Cheques and Postal Orders Accepted

Now in our Twenty Sixth Year

Further details from the Membership Secretary

John Gilpin, 181, Urmston Lane,
Stretford, Manchester, M32 9EH (UK).
Tel. +44 (0) 161 865 2872
Email: membership@quanta.org.uk

Visit the QUANTA Web Site
http://www.quanta.org.uk

```
10 REMark June 2007
20 WMON2: OPEN #7,con: COLOUR_24, #7
```

Open a centred window on channel 7 60% screen width by 50% screen height with a light blue background.

```
30 Centre_scr 7,60,50: PAPER #7, $84C6: INK #7,0: CLS #7: BORDER #7,1,255
40 LET Pic$="win2_Sbasic_Loudness_Control_bmp"
45 LET FntLength=FLEN(_FOUNTS_Helvetica_fnt)
47 LET Base=ALCHP(FntLength)
```

Replace the default QL fount with the Helvetica fount and apply to channel 7.

```
48 LBYTES win3_FOUNTS_Helvetica_fnt,Base
49 CHAR_USE #7,Base,0
50 REPeat start
60 BMP8LOAD #7, Pic$: INK #7,0
```

Load the background picture into channel 7. The image is 615 x 350 pixels.

```
170 REMark -- GET RESISTANCE --
180 REPeat GetRes
190 Blank_line 7,35,1
200 AT #7,35,1: INPUT #7;"Resistance of Potentiometer in Ohms? ";RPot
205 LET RPot=INT(RPot)
```

Limit values to between 47000 ohms and 2 million ohms.

```
210 IF RPot >46999 AND RPot <=2E6 THEN EXIT GetRes
220 AT #7,36,1: PRINT #7;"Must be greater than 46999 Ohms and less than or equal to 2 Megohms"
240 Blank_line 7,35,1
250 END REPeat GetRes
375 Blank_line 7,35,0: Blank_line 7,36,0
380 REMark -- SHOW RESULTS --
390 OVER #7,1
```

Find one third of the total potentiometer resistance (TopTap) and the value of R1 (TopRes) which is in parallel with C2.

```
450 LET Toptap=RPot*.33: LET TopRes=RPot-Toptap
```

Find one sixth of the total potentiometer resistance (BotTap) and the value of R2 (Mid) which is in parallel with C4.

```
455 LET BotTap=RPot*.167: LET Mid=Toptap-BotTap
460 LET FTop=3500: LET FBot=5000: LET w=2*PI
470 LET C2=1/(w*TopRes*FTop)*1E12
475 LET C4=1/(w*Mid*FBot)*1E12
```

Print the results for C2 and C4 rounded up to the nearest whole value with no decimal places.

```
480 AT #7,2,50: PRINT_USING #7,"C2=####. pF",C2
490 AT #7,3,50: PRINT_USING #7,"C4=####. pF",C4
495 AT #7,4,50: PRINT #7, "C1 & C3= 33 nf"
500 AT #7,5,50: PRINT #7,"Potentiometer Res.= ";RPot/1000;" Kilohms"
510 AT #7,6,50: PRINT #7, "R1= ";TopRes/1000;" Kilohms"
520 AT #7,7,50: PRINT #7, "R2= ";Mid/1000;" Kilohms"
522 AT #7,8,50: PRINT #7, "R4 & R5= 12 Kilohms"
525 AT #7,9,50: PRINT #7, "Top Tap= ";Toptap/1000;" Kilohms"
530 AT #7,10,50: PRINT #7,"Bottom Tap/R3= ";BotTap/1000;" Kilohms"
540 INK #7;$320075: AT #7,14,50: PRINT #7,"Press F5 to repeat"
560 PRINT #7; TO 50;"or F7 to Exit"
570 OVER #7,0: keyscan
580 SELect ON z
590 REMark -- F7 EXIT PROGRAM --
600 ON z=238
610 CLOSE #7: WMON2: EXIT start: CLEAR
660 IF z <>248 THEN keyscan
```

```
670 REMark -- F5 START AGAIN --
680 END SELect

690 END REPeat start
700 REMark -- WHICH KEY ARE YOU PRESSING? --
710 DEFine PROCedure keyscan
720 REPeat Scan
730 LET z=CODE (INKEY$(#7))
740 IF z=238 OR z=248 THEN EXIT Scan
750 END REPeat Scan
760 END DEFine keyscan
770 DEFine PROCedure Centre_scr (chnl,xpcnt,ypcnt)
780 WHEN ERRor
790 IF ERR_NO THEN PRINT #0, "Channel is not open": STOP: WMON2
800 IF ERR_OR THEN PRINT #0, "Width or height dimensions greater than 100%": STOP: WMON2
810 END WHEN
820 WINDOW #chnl, SCR_XLIM * (xpcnt/100),SCR_YLIM * (ypcnt/100), (SCR_XLIM-((xpcnt/100) *
    SCR_XLIM))/2,(SCR_YLIM-((ypcnt/100) * SCR_YLIM))/2
830 END DEFine Centre_scr
840 DEFine PROCedure Blank_line (chnl,d,a)
850 AT #chnl,d,a: PRINT #chnl;"   "
860 END DEFine Blank_line
```

Using the component values found in Part 1 of this article the one sixth tapping point is between switch positions 15 & 16 (counting clockwise). With a 100Kohm control one sixth of the resistance is 16666 ohms. At step 15 the total measured resistance of the chain is around 15770 ohms so to get the correct tapping point I connected an 820 ohms resistor at step 15 in series with a 3300 ohm one connected to position 16. The tap is taken from the junction of the series connected resistors. The one third tapping point is 33333 ohms which is at position 18. The chain resistance here was around 31400 ohms and in this instance I decided not to add any series resistors. Below are the values for the control.

Switch position	
23	R0 = 20567
22	R1 = 16337
21	R2 = 12977
20	R3 = 10308
19	R4 = 8187
18	R5 = 6503
17	R6 = 5166
16	R7 = 4103
15	R8 = 3259
14	R9 = 2589
13	R10 = 2056
12	R11 = 1633
11	R12 = 1297
10	R13 = 1030
9	R14 = 818
8	R15 = 650
7	R16 = 516
6	R17 = 410
5	R18 = 325
4	R19 = 258
3	R20 = 205
2	Rf = 794
1	0 (tied to ground rail)

Figure 3 shows the results of the calculations for a 100Kohm potentiometer. The completed control is shown in figure 4. Part of the resistor chain can be seen and the compensation capacitors soldered to a small piece of Veroboard. At top right is one of a pair of No. 76 triode valves used in my preamplifier.

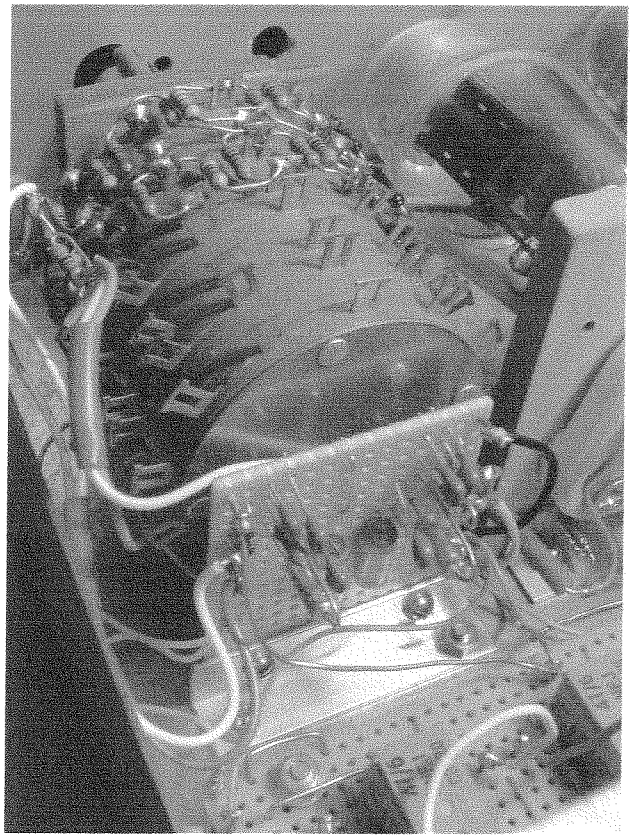
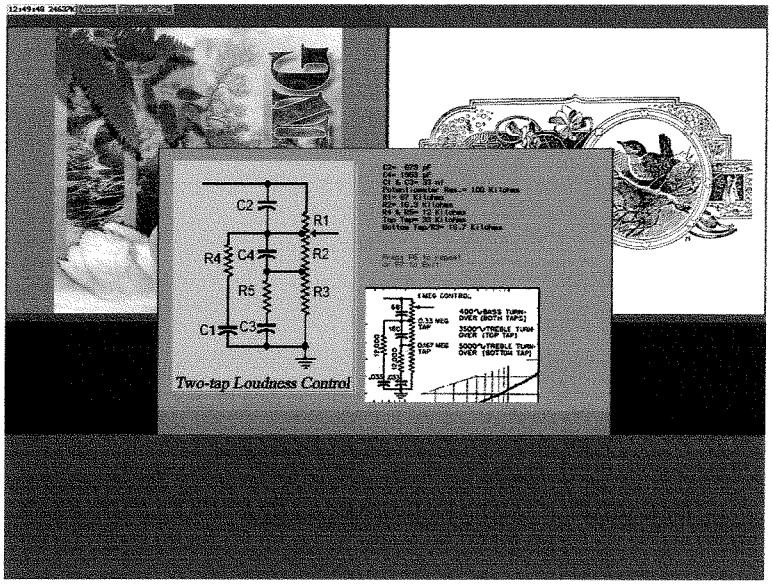


Fig. 4

It is possible to increase or decrease the level of bass and treble boost by altering some of the component values. To get more bass boost decrease the values of R4 and R5 and increase



the values of C1 and C3. The combination of components should be chosen to give the same turnover frequency (400Hz). To increase the treble boost raise the values of C2 and C4.

I have been using the control for over a year now. The effect of the control is quite subtle unlike a normal tone control, and when listening at low volume levels the overall tone content of the sound appears unchanged. At higher levels the control has a lesser and lesser effect on tone.

Fig. 3

LETTER-BOX

George Gwilt writes: **Letter to QL Today re Norman Dunbar's Article on Assembler - Part 24**

It was with some trepidation that I started reading Norman's latest article on Assembler since it was devoted to my SETW. What brickbats would be flying? In the event I think Norman let me off lightly. However, there are one or two comments which might be useful to anyone following Norman's instructions.

1. A very minor point is that the name of the zip file for EasyPEasy is peasp02.zip. (Only one s)
2. It is true that on starting SETW you need to be sure that you are going to get non relocatable output if you are going to use GWASL to assemble it. The parameter -abin will ensure that. However, this choice can also be made by configuring SETW. I checked that the version on my web site is already configured to give the right output. Hence leaving out -abin would do no harm.
3. In the section on "The Main Window" Norman advises you to read the prompt carefully before pressing ENTER because "there's no going back!" This is the first brickbat. I have myself found it annoying that you can't go back to alter the number of objects or windows. Part of my solution was to allow a quick way of aborting the process. Pressing ESC will often allow you to stop the program. Otherwise you would have to plough through the remainder of what could be a large number of questions before ending the process to start again. The other part of my solution was to make it relatively easy to alter what output is eventually produced.
4. I was amused at Norman's instructions to set the size of windows according to the little box at the bottom of the screen. I must confess that I very rarely use that and simply rely on eye to check the size of windows and position of objects. It might be useful to know that the size of the information window is constrained by SETW to be large enough to hold the biggest of its objects.
5. The width of "Hello World" is 66 pixels. Norman says it is 72. My guess is that he typed "Hello World", or possibly one of "Hello World ", and " Hello World".
6. Norman was using an operating system with GD2 colours. On a more primitive machine SETW works slightly differently. In this case, the colours for windows, borders and text items must all be chosen from the mode 4 set presented. Pressing ENTER immediately for every colour would result in them all being white! The default colours for GD2 are in fact the system palette entries set up for GD2. For example the colour in hello_asm for the border of the information window comes out as 526, or $020E, which is defined as sp.infwinbd (Information window border). This is not a colour in itself but is an entry to the system palette colours of which there are four sets. These, by default, are

set to be combinations of the QL 4, black, red, green and white, but these can be redefined by a user. When SETW is finished the window it has produced is shown with system palette number 0 being used.

7. Finally, Norman says that the result, hello_asm, appears "On ram1, in my case". Actually all output always goes to ram1_. However, SETW can be set to put any of the output files to some other directory as well. In fact the SETW on my web site is configured to send the _asm files to "win1_ass_pe_". If the file with name "win1_ass_pe_hello_asm" can't be opened, SETW will say so otherwise the information will appear there too as well as in ram1_.

Random & PI

by Stephen Poole

In one of his 'Gee Graphics' articles, Herb Schaaf asked if anyone had a routine to generate Random numbers. After unsuccessfully trawling the net, I decided to write one of my own. This proved to be much more difficult than I first thought, as any 'seed' used in an algorythm will always produce the same number sequence from that seed. So if you reset your unexpanded QL and run a program at boot time you will always get the same 'random' sequence.

By definition, a random number should give a uniform spread over its entire range and be totally unpredictable. Early methods therefore used the system clock to seed the algorythm, as there is no way of predicting when the clock will tick...So this was the method I used. All the program does is to set a FOR loop running, and interrupt it when the clock ticks. (Set the counter 'ct' to slightly more than the number of loops your machine can do in one second). Obviously this program should only be used at occasional intervals, as you can otherwise bias the results by guessing when the loop will start.

Another method used by Apple sets the clock ticking and interrupts the counter when you hit a key. Most computers now seed the random number generator at start-up, then reseed it periodically from the previous seed. Indeed the QL uses the date tick if you use RANDOMISE DATE. But the QL Randomise function contains a bug for which Mark Knight gave a fix in his QL Today 'True_Randomise' article.

My method waits for you to hit a key before producing the next random number, it is not a Formula One model. It merely demonstrates a very simple generator to program. You can see how random the output is by looking at the dispersion of the vertical lines. If you only use it every few minutes it will be very accurate!

If you want your random number to lie within a set range, scale your range to that of the computer, using a simple ratio. This takes but a few SuperBasic statements... If you find QL Integer size limits normal QL random number size, just convert the random numbers to strings and concatenate them.

The British Premium Bond generator, 'Ernie', used coupled Neon diodes to get the random winning numbers, it having been proved that this device produced a truly random 'White Noise' series when sampled. Another important use of random numbers is in nuclear warfare : The Test Ban Treaties forced countries to produce atom bomb simulators to test new modifications to weapons. These simulators need perfectly random numbers to predict the chain reaction where neutrons are ejected randomly. Some algorythms use rounded-off 'lost' digits in suitable pseudo-random calculations to provide seeds. These have the advantage of being virtually instantaneous, but are still 'somewhat' predictable. Does anybody know what method the QL uses?

Being a transcendant number, there is no known formula to calculate PI, so you could get extract 'random' numbers from the PI sequence, as long as you don't know where in that sequence you started. This 'Viete Method' uses the surface of polygons to obtain PI by iteration very efficiently. If you want a challenge, try modifying the program to produce PI to 100 decimals or more. This would of course require you to write suitable multi-precision Square-root and arithmetic routines.. (Maybe I shall do some for a forthcoming magazine). Most other routines to calculate PI are horrendously slow...

Happy Lucky Dipping

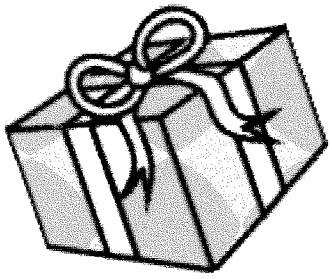
Editor: As the listing is very short, we decided to print it on the next page.

```
100 ::
110 REMark PIE_bas by S.Poole. v7june 2008.
120  CLEAR: OPEN#1,con_16: WINDOW 512,256,0,0: BORDER 1,4
130  CLS: RANDOM: CLS: WINDOW 256,206,256,0: PIE: STOP
140 :
150 DEFine PROCedure RANDOM
160  REMark Original DIY method:
180   REMark Scale is larger than highest random number:
185   REMark Ajust it to fit your screen and machine speed:
190    SCALE 128000,0,0: I$='': REMark 200000
200  REPeat loop
210     REMark When QL clock ticks stop the counter:
220      d1=DATE: FOR ct=1 TO 2E6: IF DATE<>d1: EXIT ct
230      AT 1,1: PRINT ct,: LINE ct,0 TO ct,ct
235      :
237      AT 22,1: PRINT 'Hit a key to start the counter or q to quit:'
240      I$=INKEY$(#1,-1): IF I$=='q': EXIT loop
250  END REPeat loop
260 END DEFine
270 :
280 DEFine PROCedure PIE
290  CLS: p=2: n=.5: r=SQRT(n)
300  FOR x=1 TO 12: p=p/r: r=SQRT(n+n*r): PRINT\p,x,
310 END DEFine
320 ::
```

25% EXTRA

by Geoff Wicks

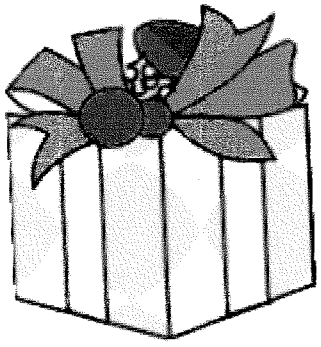
This issue of QL Today comes with the bonus of 8 extra pages for our readers. This is possible because Jochen will be able to send it from Austria where the postage costs are lower than in Germany and the Netherlands.



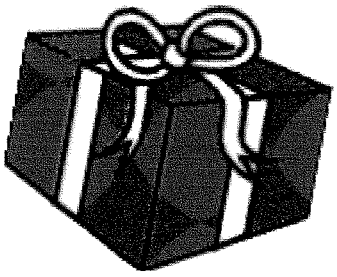
Once again much of the magazine is taken up by Tony Tebby's contributions.

Reactions to the last issue indicate that our readers are enthusiastic about these articles, which contain material that has never been published elsewhere.

The articles do, however, give us a few logistical headaches at QL Today because we want to ensure that other contributors are well represented in the magazine. For this reason we are very happy to be able to publish the extra pages in this issue.



At the moment we are trying to be fair to all our contributors to ensure they do not have to wait too long for their articles to appear. This means that some of our regular writers may not appear in individual issues of the magazine. We hope this will not put off either regular or occasional contributors writing for QL Today.



Very slowly we are now clearing up the backlog of copy and to ensure sufficient material for issues 3 and 4, we shall be pleased to receive further contributions. In particular short contributions would be very welcome, not only as a way of increasing the number of contributors per issue, but also to enable us to fill the last few pages in any one issue with worthwhile material.

The QL Show Agenda

Come to Vienna!

International QL-meeting 2010 in Prottes (near Vienna). Thursday, 3rd (bank holiday) to Sunday, 6th of June 2010.

Gerhard Plavec, the organiser, has already placed a lot of useful information on his website http://kuel.org (German, French and English)

He plans to turn the Saturday into the main day, but if the majority of visitors prefers to come on Friday, it can easily be changed.

On his website, you'll find already loads of information about accomodation (including staying at Gerhard's place), even with a tent ... he can also provide electricity for visitors who come with their motor homes. In both cases, please contact Gerhard in advance.

Prottes can be reached directly by car and train, and, of course, with every travelling method via Vienna (airport or ship, e.g. from Bratislava).

There are several tourist sites nearby (not to mention Vienna itself). The railroad museum in Strasshof (very close) will be open all four days, and they will even get steam engines going on Sunday. If there is enough interest, Gerhard may ask if they would do it on the main day (see above, most likely Saturday or Friday).

Gerhard can be contacted via email: gerhard.plavec@gmx.at or phone +43 699 81856765

We are informing you early to ensure you can fit the meeting in with other holidays or journeys ... so let's turn this event into a big event, kind of re-union of QLers who have not met for a long time!

The Next Issue

We plan to have the next issue ready for you towards the middle of March.
As always, it depends on how quickly we get reviews, articles etc.
The more material we get and the sooner we get it, the quicker the next issue will be in your hands, and the better it will be. We depend on your support.

MERRY XMAS AND ALL THE BEST FOR 2010 FROM THE QL TODAY TEAM